Form Approved
OMB No. 0704-0188

1. AGENC	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 3, 1992	Final Report 12/87 - 8/92

12

4. TITLE AND SUBTITLE

Knowledge and Processes in Design

5. FUNDING NUMBERS

C·N00014-88-K-0233
PE·61153N
PR·RR04206 TA·RR04206-01
WU·NR4422550

6. AUTHOR(S)

Peter Pirolli

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Graduate School of Education
Education: Mathematics, Science, & Technology
University of California at Berkeley
Berkeley, CA 94720

8. PERFORMING ORGANIZATION REPORT NUMBER

DPS Final Report

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Cognitive Science Program
Office of Naval Research (code 1142 PT)
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Generic design, problem solving, Soar	**170**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	**Unclassified**	**Unclassified**	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

SEPTEMBER 1992 DPS FINAL REPORT

Knowledge and Processes in Design

Peter Pirolli
University of California at Berkeley

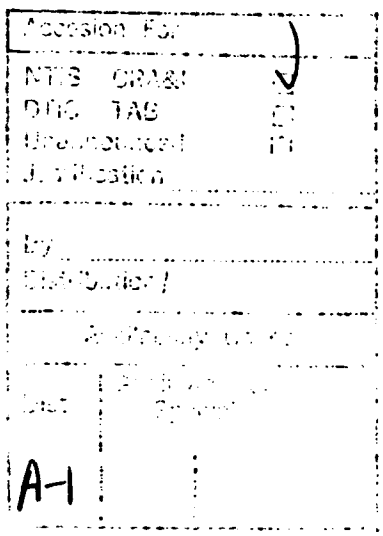
Portions of this research were funded by the Cognitive Sciences Program, Office of Naval Research under contract N00014-88-K-0233, Contract Authority Identification Number NR4422550. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

Knowledge and Processes in Design: Final Report

Office of Naval Research Contract N00014-88-K0233
Cognitive and Neural Sciences Division

Peter Pirolli
University of California, Berkeley

August 1992



1. Introduction

Our project concerned information-processing characterizations of the knowledge and processes involved in design. Design is a complex ill-structured task. By their very nature, such ill-structured tasks involve processes of finding, representing, refining, and reformulating goals and constraints, generating appropriate problem spaces to work in, retrieving, integrating, and evaluating information from long-term memory or external sources, and managing complex assemblies of past and current decisions (Mostow, 1985; Reitman, 1964; Simon, 1973).

In classic theories of problem solving in psychology, a problem solver is faced with a problem situation and develops a representation for the problem (problem structuring). In classical terms, this representation may include information about the current state of the problem situation, a goal, and possible sets of problem solving actions (operators) that may be relevant to changing the current state to conform to the desired goal. A particular goal, a set of states, and a set of operators defines a problem space. Problem solving is characterized as a search process in a problem space, in which the problem solver proposes and decides upon possible operators and sequences of operators that will transform the current state into a goal state. Much of the research on the psychology of problem solving has focused on understanding the knowledge and processes involved in solving well-structured, but semantically impoverished, puzzle tasks. In these cases, the initial problem state, the goal state, and the problem solving operators are well-defined, and problem solving difficulties are largely due to the complexities of searching for and finding the right sequence of problem solving actions. It is only recently that cognitive psychology has started to examine in detail the kinds of problem solving and decision making that occur in ill-structured problems such as those that are typical in expert design.

Over the course of our project we made substantial progress in a number of areas. We collected and analyzed verbal protocols from instructional designers, architects, and mechanical engineers. We developed a framework for characterizing the problem spaces of design that generalizes across design tasks, that is based on task invariants, the invariants of the human information processor, and the logical structure of design. We developed a protocol analysis formalism that can be used to analyze protocol data from experts in different disciplines, and more importantly, can be used to connect statement-by-statement codings to more abstract characterizations of design. This analytic technique has been used to analyze and compare design in three disparate design tasks. The data, methodology, framework, and theory are presented in detail in (Goel & Pirolli, 1989; Goel & Pirolli, in press) and included here as Attachments 1 and 2. In addition, we developed a computer simulation, implemented in Soar, of one of our design subjects, which is presented in (Pirolli & Berger, 1991), included here as Attachment 3. This simulation served to provide a precise sufficiency test of our analysis. The flip side is that the simulation forced us to provide a precise description of the cognition of one of our designers.

Many interesting phenomena arise in design problem solving, and one of these is the extensive use of external symbol systems. In developing an analysis of the relationship between problem solving and external symbol systems, a theoretical framework for the analysis was developed to categorize semantic and syntactic properties of external symbol systems. This framework (largely worked out by Vinod Goel) is presented in (Goel, 1991a, 1991b). An experiment was developed to test hypotheses motivated by this theoretical analysis, largely focusing on the relation of the ambiguity and density of symbol systems to different phases of problem solving (roughly, why is it that sketches are

dominant during early design work but precise notations dominate later problem solving). This empirical work is presented in (Goel, 1992), included here as Attachment 4.

These accomplishments are important for several reasons. Cognitive science has made substantial progress in the study of well-structured or semantically rich problem solving. However, the study of ill-structured problem solving, such as design, has received less attention. Furthermore, approaches taken to design have concentrated on individual tasks, rather than characterizations that generalize across design tasks. Finally, there has been very little work that attempts to relate properties of external representations to different phases of problem solving. Our work suggests that we might be in a position to create an integrated framework that:

- Provides middle-level nomothetic theory, allowing us to specify the universals of design across tasks and disciplines. Thus, the framework can be used to develop *normative* analyses and models across subjects.
- Provides a precise framework and language for *idiographic* theory and analyses, allowing us to analyze individual cases of design. Thus, the framework can be used to develop *descriptive* analyses and models of individual subjects.
- Provides computational theory, allowing us to guarantee that our models are *sufficient* to describe individual behavioral patterns.
- Describes *the role of external representations in problem solving*. Although some recent work in cognition has begun to on this aspect of problem solving (e.g., Larkin & Simon, 1987) these have tended to focus on well-structured representations for well-structured problem solving tasks. Our focus has been on the generative role of representations in problem solving.

Additional progress in the study of design could lead to progress in the study of other ill-structured tasks, such as writing, strategic analysis, formulation and revision, and other tasks usually thought to involve creativity and expertise.

2. Formalizing the Notion of Generic Design

Conceptually, one of our initial moves in studying design involved developing an analytic framework that discriminated design tasks from other kinds of problem solving, and generalized across tasks that would generally be called design. Our initial report in this area was presented in a paper by Goel and Pirolli (1989; see Attachment 1). In that paper, we presented the *design problem space*, which provides a characterization of the abstract structure of generic design tasks. Our first move in developing the design problem space was the observation that design is a "natural kind" of problem solving (Goel & Pirolli, 1989):

> Design is too complex an activity to be captured in a one-line definition...our characterization of design starts with the observation that design is a category that exhibits what Rosch calls prototype effects. (Goel & Pirolli, 1989, p. 23).

Our next step was to realize that the abstract structure of the design problem space would be, in part, determined by constraints resulting from the invariants of the task environments across design situations, combined with the invariants of human information

processing. This involved an analysis of the structure of the prototypical design tasks, the identification of the substantive invariants of the design task environment and human information processing system, and the ways that these relate to the structure of the design problem space (Goel and Pirolli, 1989)

Characterizing design at this level of abstraction has both costs and benefits. On the one hand, it moves us away from the study of individual tasks to more general and abstract characterizations of the intricate and intertwined interactions of the human problem solver and a task environment. Such a move is important if we seek to gain a general understanding of the nature of such complex problem solving. On the other hand, such a level of description is several levels removed from the typical formalisms used to describe verbal behavior at the grain size of individual statements. A substantial amount of our efforts have been devoted to elaborating the design problem space analysis simultaneous with our development of a statement-by-statement verbal protocol coding scheme with the aim of developing a multi-level coherent analytic methodology.

3. Empirical Investigations

Two other important developments in our research have been to develop design tasks that are effective for studying design, and to develop a comprehensive data analysis methodology. From pilot studies we realized that we needed to study expert designers working on tasks that are novel, but not beyond their expertise. We collected data from architects, mechanical engineers, and conducted a larger experiment with instructional designers.

We studied these experts on three sorts of design tasks appropriate to subjects' expertise. Our most extensive data collection has concerned instructional design, in which design briefs were presented to 10 professionals who work for an international office systems company. The problems involved the design of instruction for a word processing system familiar to the subjects. The problems were varied with respect to the target audience, the instructional medium, and the nature of the desired specifications. Our architecture task concerned the design of the site and building of an automated postal teller system on the UC Berkeley campus. The mechanical engineering task involved the design of the electro-mechanical system for an automated postal teller system. Each task took about two to four hours to complete. Each designer worked alone, while thinking out loud, using only pencil and paper.

Goel and Pirolli (in press; see Attachment 2) developed a hierarchical protocol coding scheme to analyze our protocol data. At the finest grain of analysis the coding scheme captured the operation being performed, the propositional content of the statement, the knowledge source on which the content was based, and the context of the statement within larger processing events in the protocol. These statement-by-statement codings were then organized into larger control-flow structures centered around design components called *modules*. The general assumption was that the cognitive representation of the designed artifact had a nearly-decomposable structure. Each major component of this decomposition was called a module, and within modules there may have been submodules.

In our formulation of the design problem space, we (Goel & Pirolli, 1989) proposed that control flow is organized around *leaky modules*. That is, design is organized into chunks of activity centering on individual modules. However, because modules are in fact interrelated, the design of individual modules must occasionally involve functional level assumptions about related modules, or occasionally involve putting the design of the

current module on hold and attending to a related module. Goel & Pirolli (in press) presented protocol analyses that illustrated leaky modules and bursts of activity in which a particular module or submodule was the focus of attention. Within these units, other modules may have been temporarily attended to or mentioned. This leads to what we called a *limited commitment mode control strategy*.

Goel and Pirolli (in press) also presented various analyses examining how the focus of design shifts through larger-grained identifiable phases of problem structuring and problem solving. The analyses examine (a)the aspects of the design being considered, (b) the sources of knowledge for design elements and decisions, (c) degree of commtment to decisions, and (d) the distribution of operator activity.

4. Simulation Work

We selected a protocol of an instructional designer for our simulation efforts. This particular protocol was chosen because (a) the subject was highly articulate and verbal, (b) the subject was one of the most productive of the instructional designers studied (he completed a draft of the instructional text including formatting and diagrams), and (c) this subject produced copious scratch notes and outlines indicating his developing design. The task for this designer was to design instruction on the Viewpoint text editor for an office of 10 secretaries who had some experience with MacIntoshes. The instruction was to be a set of six independent-study, stand-alone modules, delivered by text. This designer was instructed to produce a general outline of the sessions and to specify the first session is as much detail as possible. The simulation was implemented in the Soar (Newell, 1990) production system and is reported in (Pirolli & Berger, 1991) and in Attachment 3.

The main goal of our simulation was to generate the same written output, in the same order, as the designer, and to be consistent with the verbal protocol. The simulation operates in multiple problem spaces (as is typical in Soar programs). These included:

- A design-instruction problem space in which the task is implemented. The states in the space include a representation of the task brief, the paper output (e.g., the design sketches), and a mental model of the design artifact. The operator specify-instructional-plan is selected in this space to solve global design tasks.

- When there is insufficient knowledge to directly implement the specify-instructional-plan operator, an impasse occurs and the operator is implemented in its own specify-instructional-plan problem space. In order to specify the instructional plan, the plan is formulated and written using the operator formulate-instructional-plan.

- If the formulate-instructional-plan operator hits an impasse, it is implemented in its own problem-spaces. The formulate-instructional-plan space involves operators that specify the instructional objectives, the target population, the task and content analysis, and the subplans for the individual instructional sessions. Each of these areas is a module, as discussed earlier.

- The task and content analyses are carried out by an analysis of the Viewpoint manual table of contents. An analyze-contents space carries out the task of reading through the hierarchical table of contents, elaborating and evaluating each item, and deciding whether to include the item in the instruction.

- Task-specific knowledge associated with the `specify-instructional-plan` space organizes instructional transactions. Impasses on operators working out the design of these transactions lead to specialized problem spaces including: (a) a determine-course-relevance space that fills out a schematic structure indicating why course content is relevant to students, (b) a determine-course-sequence space that decides how to allocate instructional content into the available sessions, and to sequence the sessions, (c) a determine-lesson-instruction problem space that is used to refine the instructional transactions for a lesson.

Our simulation is consistent with our protocol analysis scheme in several respects. Most importantly, the problem spaces are generally organized around modules (e.g., the target population, the content analysis), and has the characteristics of a limited commitment mode strategy (e.g., proposing, elaborating, evaluating, then deciding). The simulation uses about 300 production rules and completes the task in 1050 decision cycles. Assuming that a decision cycle simulates human problem solving at the order of magnitude of 10 secs, the 1050 decision cycles is of about the same scale as the 3 hours of problem solving engaged in by this subject.

5. External Representations and Design

One of our observations about design was that different sorts of external representations seemed to be correlated with different phases of design. Vinod Goel addressed this issue in (Goel, 1992), included here as Attachment 4. Hypotheses about the relation between design problem solving and external representations centered on the *ambiguity* and *density* of representational systems (Goel, 1991a). The early phases of design involve higher amounts of the generation of new or alternative design elements than later phases. Later phases of design involve higher amounts of refinement of existing design elements. Representation systems of high ambiguity and high density are expected during the early phases of design. Symbols in such systems are more easily manipulated to produce novel referents (new ideas). Free-hand sketches are an example of such representational systems. Drafting -type representational systems have less ambiguous semantics and lower density.

Goel (1992) performed a study of industrial and graphics designers working on a probem in which they were restricted to either sketch-type diagrams or drafting-type diagrams during either the early or late phases. The general structure of design problem solving did not seem to be affected by choice of representational system, in terms of overall length of the design sessions or the number or duration of episodes demarcated by attention to particular design elements. However, when designer were allowed to use free-hand sketches they produced more alternative design elements than designers restricted to drafting-type systems. Free-hand sketches were also associated with more episodes of *reinterpretation*, in which the meaning of a diagram suddenly changed. This occurs when, for example, a subject draws a light bulb and then recognizes it as a human head, which in turn leads to a new design idea.

These results highlight the generative role of diagrammatic representations in problem solving. The standard view of the role of diagrams in problem solving (e.g., Larkin & Simon, 1987) is that they serve as data structures that have particular cost characteristics for storing or accessing information. The work of Goel (1992) illustrates how diagrams can support the creative generation of ideas and that this property is related to specific features of representational systems.

References

Goel, V. (1991a). *"Ill-structured" representations for ill-structured problems* (Tech. Rep. DPS-4). Berkeley, CA: University of California, School of Education.

Goel, V. (1991b). Notationality and the information processing mind. *Minds and Machines, 1*, 129-165.

Goel, V. (1992). *The cognitive role of sketching in problem solving* (Tech. Rep. DPS-7). Berkeley, CA: University of California, School of Education.

Goel, V., & Pirolli, P. (1989). Motivating the notion of generic design within information-processing theory: The design problem space. *AI Magazine, 10*, 19-36.

Goel, V., & Pirolli, P. (in press). The structure of design problem spaces. *Cognitive Science*, ,

Larkin, J., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science, 11*, 65-99.

Mostow, J. (1985). Toward better models of the design process. *AI Magazine, 6*, 44-57.

Newell, A. (1990). *Unified theories of cognition* . Cambridge, MA: Harvard University Press.

Pirolli, P., & Berger, D. (1991). *The structure of ill-structured problem solving in instructional design* (Tech. Rep. DPS-5). Berkeley, CA: University of California, School of Education.

Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelly, & G. L. Bryan (Ed.), *Human judgements and optimality*. New York: Wiley & Sons.

Simon, H. A. (1973). The structure of ill-structured problems. *Artificial Intelligence, 4*, 181-204.

ATTACHMENT 1

FEBRUARY 1989 REPORT NO. DPS-1

Motivating the Notion of Generic Design Within Information Processing Theory: The Design Problem Space

Vinod Goel and Peter Pirolli

University of California at Berkeley

In: *AI Magazine,* **10,** 19-36, (1989)

Portions of this research were funded by the Cognitive Science Program, Office of Naval Research under contract N00014-88-K-0233, Contract Authority Identification Number NR4422550. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

MOTIVATING THE NOTION OF GENERIC DESIGN WITHIN INFORMATION PROCESSING THEORY: THE DESIGN PROBLEM SPACE†

Vinod Goel
Department of Architecture
&
Peter Pirolli
School of Education
University of California at Berkeley

ABSTRACT

The notion of generic design, while it has been around for 25 years is not often articulated, especially within Newell and Simon's (1972) Information Processing Theory framework. Design is merely lumped in with other forms of problem solving activity. Intuitively one feels that there should be a level of description of the phenomenon which refines this broad classification by further distinguishing between design and non-design problem solving. However, Information Processing Theory does not facilitate such problem classification. This paper makes a preliminary attempt to differentiate design problem solving from non-design problem solving by identifying major invariants in the *design problem space*.

There are four major steps in the strategy: (1) characterize design as a radial category and flesh out the task environment of the central or prototypical cases; (2) take the design task environment seriously; (3) explicate the impact of this task environment on the design problem space, and (4) argue that, given the structure of the information processing system as a constant, the features noted in the problem spaces of design tasks will not all occur in problem spaces where the task environment is vastly different. This analysis leads to the claim that these features are invariants in the problem spaces of design situations, and collectively constitute a design problem space.

Descriptive protocol studies are used to explore the problem spaces of three prototypical design tasks from the discipline of architecture, mechanical engineering, and instructional design. The following eight significant invariants are identified: (A) extensive problem structuring, (B) extensive performance modeling, (C) personalized/institutionalized evaluation functions and stopping rules, (D) a limited commitment mode control strategy with nested evaluation cycles, (E) making and propagating commitments, (F) solution decomposition into leaky modules, (G) role of abstractions in the transformation of goals to artifact specifications, and (H) use of artificial symbol systems. The paper concludes by drawing some morals for the development of computer-aided design (CAD) systems, noting the limitations in the work, and indicating directions for further research.

---

† The authors are indebted to Dan Berger and Susan Newman for much useful discussion and argumentation on the contents of this paper. Vinod Goel gratefully acknowledges the support of the following sources of funding during the course of this work: an Andrew Mellon Fellowship, the Myrtle L. Judkins Memorial Scholarship, a Canada Mortgage and Housing Corporation Scholarship, and a summer internship at the Institute of Research on Learning at Xerox Parc. Portions of this research were also funded to Peter Pirolli by the Office of Naval Research under contract N0014-88-0233.
Correspondence should be directed to Vinod Goel at the Institute of Cognitive Studies (Bldg T-4), University of California at Berkeley, Berkeley, CA 94720.

1. INTRODUCTION

The term *generic design* denotes two related ideas. It suggests that design as an activity has a distinct conceptual and cognitive realization from non-design activities, and that it can be abstracted away from the particulars of the knowledge base of a specific task or discipline and studied in its own right. It has its origins in the design methodology research of the 1960s (Cross, 1986). At this time the observation was made that the various design methods, while they differed in particulars, shared a common pool of assumptions which conceived the design process as moving through the following sequence of steps: (i) an exploration and decomposition of the problem (i.e., analysis); (ii) an identification of the interconnections between the components; (iii) the solution of the subproblems in isolation; and finally, (iv) the combination (taking into account the interconnections) of the partial solutions into the problem solution (i.e., synthesis). On the basis of this observation many researchers concluded that "the logical nature of the act of designing is largely independent of the character of the thing designed" (Archer, 1969, p.76). However, they did not go on to develop the concept to any significant extent.

Subsequently these assumptions were questioned by other researchers (Akin, 1979; Lawson, 1979) working in the different framework of Newell and Simon's (1972) Information Processing Theory (IPT).[1] While the concern of the earlier researchers was with the development of "systematic design methods" to help designers (often working in teams) to deal with the increasing amount and complexity of project information (Cross, 1986), the concern of the latter is with explicating the internal structures and procedures individual cognitive systems use during design activity, with what Eastman (1969) called *intuitive design*.

The study of intuitive design, within an IPT framework, has become a dominant mode of research into design activity.[2] But this research is moving in two directions which are rather dissatisfying from the perspective of developing a cognitive theory of design. First, the research tends to be discipline-specific and even task-specific (e.g., Kant and Newell, 1984; Kant 1985; Steier and Kant, 1985; Jeffries et al. 1981; Ullman et al., 1986; Akin, 1979, 1986). Second, there is a proliferation of disciplines and activities being labeled as "design." Thus for example, Perkins (1986) labels the process of knowledge acquisition "design." Thomas (1978) analyses communication as a design process. Thomas and Carroll (1979) assume that letter writing, naming, and scheduling are all design activities. The first of these trends flies in the face of the intuition lying behind the notion of *generic* design. The second threatens to drain the word *design* of all meaning.

One reason for these trends is the nature of IPT itself. Within IPT design is problem-solving activity. But problem solving encompasses a wide range of cognitive activity; indeed, according to some theoreticians, all of symbolic cognitive activity (e.g., Newell, 1980). Intuitively one feels that

[1] This discussion assumes considerable familiarity with IPT as presented by Newell and Simon (1972). The uninitiated reader is well referred to this original work.

[2] There are two major reasons for this. First there is a realization by industry that the development of effective CAD tools requires a model of the user's (designer's) cognitive processes (Ballay et al. 1984). Second there is the hope that the study of human designers will lead to insights into the automation of the design process (Kant and Newell, 1984).

there must be a description of design problem-solving activity which both captures the similarities in the problem-solving process across the various design disciplines and also recognizes the differences between design and non-design problem solving. This is the level which the term *generic design* informally tries to characterize. In the vocabulary of IPT, there must exist a *design problem space*—a problem space with major invariant characteristics across all design situations. However, as has been observed by a number of researchers (e.g., Greeno, 1978), the theory does not easily facilitate such classification. We see three interrelated reasons for this "shortcoming."

1) In some ways the vocabulary provided by IPT seems to be missing a layer. At the top level of the theory one can talk about Information Processing Systems, Task Environments, and Problem Spaces. But the next level down takes one directly to the implementation details of specific programs where one must talk about states and transformations at the level of the elementary information processes. Differentiation of problem types is readily possible only at this lower level. There is a gap in the middle where one intuitively feels there should be several intermediate levels of psychologically interesting concepts—such as generic design.

2) The structure of the information processing system is underdeveloped. Except for the size of short-term memory (STM) and read/write times, it does not impose many significant constraints on the problem space. Thus the problem space tends to be substantially task determined.

3) But the notion of task environment has not been fully explored and exploited within the theory. While the theory does say that the task environment consists of (i) the goal or desire to solve the problem, (ii) the problem statement, and (iii) *any other relevant external factors*, the fact remains that historically, the goal or motivation of the problem solver has simply been assumed, and the "other relevant external factors" have been effectively ignored.[3] The emphasis has been on how the problem statement gets mapped onto the problem space.

Within IPT there are two possible sources of invariants on the design problem space. One is the *structure of the task environment*, the other is the *structure of the information processing system* (IPS). One way of motivating a DPS is to identify task environments and information processing structures that are particular to design situations. This is precisely the strategy that will be pursued here. However, we will have little new to say about the structure of the IPS. Most of the paper is concerned with explicating the structure of the design task environment and specifying its impact on the DPS.

Organization and Overview of Paper: In this paper we would like to play out the intuition which says that the design problem space is an interesting and "natural" categorization of problem spaces. Our strategy will be to (i) characterize design as a radial category and flesh out the task environment of the central or prototypical cases; (ii) take the design task environment seriously; (iii) explicate the

[3] This is undoubtedly due to the influence of game-playing problems on which the theory grew up.

impact of the structure of the task environment and the structure of the IPS on the problem space of subjects from three different design disciplines; (iv) suggest that the features noted in these problem spaces will not all occur in a problem space where the task environment is vastly different; (v) claim that these features are invariants in the problem space of design situations and collectively constitute a design problem space. Two aspects of our strategy differentiate this work from much of the current research in design: (1) we take the structure of the design task environment very seriously, and (2) we examine data from three different design disciplines. The paper concludes by drawing some lessons for the development of CAD systems, noting some methodological limitations, and suggesting avenues for further research. We begin by characterizing design and the design task environment.

2. CHARACTERIZING DESIGN & THE DESIGN TASK ENVIRONMENT

In this section we would like to claim that design is *not* a ubiquitous activity. We no more design all the time than we read all the time, play chess all the time or engage in scientific research all the time. But the characterizations of design in the cognitive science literature would have us believe that most of us do engage in design activity most of the time. We briefly review some of this literature and conclude by offering our own, rather different, analysis.

Perhaps the most encompassing characterization of design is due to Simon (1981, p.130):

> Everyone designs who devises courses of action aimed at changing existing situations *into preferred ones.... The intellectual activity that produces material artifacts is no* different fundamentally from the one that prescribes remedies for a sick patient or the one that devises a new sales plan for a company or a social welfare policy for a state.

On this account, anyone dissatisfied with existing states of affairs and attempting to transform them into "preferred ones" is engaged in design activity. The domain of design would seem to be coextensive with the domain of problem solving.[4]

An early attempt at circumscription is due to Reitman (1964). In a paper on ill-defined problems, Reitman (1964) suggested a categorization of problems into six types based upon the distribution of information within a *problem vector*. A problem vector is a tuple of the form [A, B, =>], where components A and B represent the start and terminal states respectively, and the component => denotes some transformation function. Reitman's Type2 problems correspond to our intuitive notion of design. Typical Type2 problem statements are:

compose a fugue
design a vehicle that flies
write a short story
design a building
make a paper airplane

While these statements encompass widely varying activities, Reitman observed that they constitute

[4] Which in turn is coextensive with thinking in IPT.

formally similar problems by virtue of the amount and distribution of information among the three components of the problem vector. In the case of the Type2 or design problems, the invariant characteristic is the lack of information. For instance:

(i) The start state A is unspecified (e.g., design a vehicle...with what? putty? cardboard? prefabricated parts from GM?).

(ii) The goal state B is incompletely specified (e.g., how long should a story be? what should the plot be? how should it end?).

(iii) The transformation function => is unspecified (e.g., how should the airplane be made? by folding the paper? by cutting and pasting?).

After this seminal paper design problems became identified with ill-defined problems.

Continuing the investigation of ill-defined problems, Simon (1973) argued that problems in the world do not come prelabeled as "well-defined" or "ill-defined." Furthermore, according to Simon, "well-defined" and "ill-defined" are not mutually exclusive categories. They constitute a continuum. Where a given problem falls on this continuum is a function of the *stance* the problem solver takes to the problem. That is, the problem solver may ignore existing information or supply missing information from long-term memory or external aids. The conclusion that follows from Simon's discussion is that what constitutes a design problem is determined by the intentions and attitudes of the problem solver. This is an interesting position that has found some acceptance in the literature (e.g., Thomas and Carroll, 1979). It does however, have the effect of once again opening up the flood gates as to what constitutes design activity.

Each of these attempts at delimiting or characterizing design is due to cognitive science researchers. Designers typically offer very different definitions. A rather well-accepted one among designers is due to Eastman (1981, p.13): "Design is the specification of an artifact that both achieves desired performances and is realizable with high degrees of confidence." This statement emphasizes that the product of design is an artifact specification, and that considerations of performance and realizability are integral to the process.

While each of these definitions is interesting in its own right and has a role to play in our understanding of design, none of them is sufficient for our purposes here. Design is too complex an activity to be captured in a one-line definition; particularly a one-liner that purports to specify necessary and sufficient conditions. As such, our characterization of design starts with the observation that design as a category exhibits what Rosch (in Lakoff, 1987) calls "prototype effects." Furthermore, it is what Lakoff (1987) calls a radial category—a category in which there is a central, ideal or prototypical case and then some unpredictable but motivated variations. On this assumption, if one shows people a list of professions—e.g., medicine, legal work, architecture, teaching, engineering, and research—and asks them which are the *best examples* of design professions, they will all invariably and consistently pick out the same few cases. In this list we believe the "best examples" would be architecture and engineering. We propose to call these "good" or "central" or "prototypical" examples of design professions.

Having made this observation, we propose to take a serious look at the task environment of these prototypical design professions. In so doing we will be using the term "task environment" much more broadly than it is generally construed in IPT. We want to use it to encompass much of what is relevant and external to the problem space and the information processing system. The danger with this move is that either it results in a theoretically uninteresting term—because in some sense everything is relevant—or one is obliged to say what matters and what doesn't. We go the latter route and attempt to specify some of the more important aspects of the design task environment.

Fig. 1 approx here

The structure of the design task environment as we construe it is depicted in Fig. 1. As a first approximation one can note the following overt features:[5]

1) There are many degrees of freedom in the problem statement. (This is just a positive reformulation of Reitman's (1964) earlier point about a lack of information in design problem statements.)

2) There is delayed/limited feedback (on the order of many hours to many months) from the world during problem solving.

3) The input to the design process substantially (though not completely) consists of goals and intentions. The output is a specification of an artifact.

4) The artifact must function independently of the designer.

5) There is a temporal separation between the specification and delivery of the artifact (with specification preceding delivery).

6) There are costs associated with each and every action in the world. (i.e., There are penalties for being wrong.)

7) There are no right or wrong answers, only better and worse ones.

8) The problems tend to be large and complex.

We claim that these are significant invariants in the task environments of prototypical design situations, and we can use them as a template to identify other cases of design. To the extent that the task environment of a given problem situation meets or conforms to this template, that problem situation is a good or prototypical example of a design situation. To the extent that a task environment varies from this template—by omission of one or more of the requirements—to that extent it is a less central case of design activity.

---

[5] No attempt is being made to be exhaustive

Some problem-solving situations which fit well into the schema are instructional design, interior design, "text book cases" of software design, and music composition. Some tasks that deviate slightly are writing and painting. Here there is usually no separation between design and delivery. The problem solver actually constructs the artifact rather than specifying it. Some activities that deviate more radically are classroom teaching, spontaneous conversation, and game playing.

Note that we are not stipulating what is and is not a design activity. To do this we would have to insist that the eight task environment characteristics, or some subset of them, constitute necessary and sufficient conditions for design activity. We make no such claim. Rather, all we are suggesting is that we have here a template of some salient characteristics common to the task environment of problem situations that are consistently recognized by people as good examples of design activity. Problem situations in which the task environment fails to conform to this template on one or more accounts are deviations from the central case. In this paper we are only interested in central cases and thus have no interest in saying how far one can deviate from the prototype and still be "really" designing. Thus, we will use the label *design* to refer to situations that closely conform to the prototypical or central cases.

There are two reasons why the above might be a reasonable characterization of design for our purposes. First, it is descriptive. We look at the task environment of *some* designers and try to take it seriously. The task environment of an activity is usually overtly visible with minimal theoretical commitments (though it does require some immersion in the activity and the ability to specify the more relevant factors). Second, in IPT the structure of the information processing system is relatively underdeveloped, leaving the task environment as the major tool/resource for structuring the problem space. Furthermore, the theory asserts that people "are severely stimulus-bound" (Hayes and Simon, 1974, p.197) with respect to representation and construct a naive/transparent model of the problem based upon "the surface features of the external environment..." (Newell, 1980, p.714). Thus given the accessibility and the importance of the task environment to IPT, it seems like a good basis for classification. In the next section we examine each of the invariant features of the design task environment and hypothesize about their impact on the DPS.

3. A CASE FOR GENERIC DESIGN: THE DESIGN PROBLEM SPACE

In the previous section we identified eight interesting invariants in the structure of the design task environment. These invariants are external features of design activity that have been noted by various researchers at various times and places in the design methodology literature. But we are unaware of any studies in the IPT literature in which these factors are taken seriously and their cognitive implications sketched out. We undertake this task in this section.

Our strategy is to examine a number of designers at work and (i) reconstruct their problem space; (ii) make an "explanatory connection" between the features evident in their problem spaces and the above noted invariants of the design task environment (DTE); and (iii) make the standard argument that the problem space is as it is because of the structure of the DTE and the structure of the IPS. This last point implies that, taking the structure of the IPS as a constant, the features noted in the

problem spaces of these tasks will not all occur in a problem space where the task environment is vastly different. This leads to the claim that these features are invariants in the problem spaces of design situations, and collectively constitute a Design Problem Space. We are actually able to identify eight interesting invariants in the problem spaces of three different design disciplines. To anticipate and overview, we will claim the following:

A) The many degrees of freedom in design problem statements entail extensive problem structuring.[6] (section 3.1)

B) The delayed/limited feedback from the environment, coupled with the cost of action, and the independent functioning requirement on the artifact entails extensive performance modeling of the artifact in the problem space. This modeling is made possible by the fact that there is a temporal separation of specification and delivery phases. (section 3.2)

C) The fact that there are no right or wrong answers to design problems entails the use of personalized evaluation functions and stopping rules. (section 3.3)

D) The requirements of extensive performance modeling, along with the constraints of sequential processing and short-term memory (STM) capacity entail a limited commitment mode control strategy with nested evaluation loops. This strategy is enabled by the temporal separation of specification and delivery. (section 3.4)

E) The necessity of having to specify an artifact means that designers must make and propagate commitments. There is a tension between the limited commitment mode control strategy and the need to make commitments. (section 3.5)

F) The size and complexity of design problems combined with the limited capacity of STM require solution decomposition. However, the decomposition is not complete. The modules are "leaky." (section 3.6)

G) A phenomenon closely related to solution decomposition is the mediation of goal and artifact by abstraction hierarchies. It is entailed by the complexity of the problem, STM capacity, and the fact that the input to the design process substantially consists of goal statements while the output is an artifact specification. It is also related to the phenomenon of personalized/institutionalized stopping rules and the making and propagating of commitments. (section 3.7)

H) The last problem space invariant we note and discuss is the use of artificial symbol systems. It is entailed by the limitations on the expressive power of the "language of thought," STM capacity, sequential processing, and problem complexity. It is related to and has consequences for the phenomenon of solution decomposition, abstraction hierarchies, the making and propagating of commitments, and performance modeling. (section 3.8)

---

[6] By the use of the terms *entail* and *necessitate*, we are throughout making contingent causal claims, not logical necessary claims.

All these invariants, their interconnections, and their connections to the invariants of the DTE and the information processing system are explicated in the diagram in Fig. 2. While no claim of completeness is made for this list, it is our contention that collectively these invariants differentiate design problem spaces from non-design problem spaces. But before actually presenting and discussing each one, a word about methodology is in order.

Fig. 2 approx here

Method: The method of investigation adapted here is that of protocol analysis (Ericson and Simon, 1984). The data base consists of 12 protocols from 3 different design disciplines—architecture, mechanical engineering and instructional design. To illustrate and substantiate our claims for the purpose of this paper we will draw upon one protocol from each of the three disciplines. The decision as to which three of the protocols to use was made as follows: In the case of mechanical engineering, there was only one protocol. There were multiple protocols for instructional design and architecture. The decision among them was made on the basis of the completeness of the artifact specification and the fluency of the verbalization.

Task Descriptions: The architecture task involved the design of an automated post office (where postal tellers are replaced by automated postal teller machines) for a site on the UC-Berkeley campus. The mechanical engineering task was to design the automated postal teller machines (APTM) for the post office. The instructional design task was unrelated. It called for the design of some stand-alone text based instruction to prepare the secretaries of a medium-sized company for a transition from typewriters to the Viewpoint[7] computer environment. In each case, the subjects were given a design brief which stated the client's requirements and encouraged to probe the experimenter for further information and clarification. They were asked to "talk aloud" as they proceeded with the task. The sessions were taped on a video recorder.

Each of the tasks are complex, real world problems requiring on the order of weeks to months for a complete specification of the artifacts. We asked the architecture and mechanical engineering subjects to restrict their sessions to approximately 2 hours and gave the instructional designers approximately 3 hours. As a result, we received solutions specified to an incomplete level of detail.

Subjects: Each of the 3 subjects volunteered to participate in the study. The architect (Subject-A) is a Ph.D. student in the Department of Architecture at UC-Berkeley. He has had six years of professional experience. The mechanical engineer (Subject-M) is a Ph.D. student in the Department of Mechanical Engineering at Stanford University. His professional experience is more limited, but it has included the design of automated bank teller machines. The instructional designer (Subject-I) is a

[7] Viewpoint is an icon-based computer environment for Xerox Stars. It supports such functions as electronic mail, filing, word processing, and graphics.

professional with over 10 years experience in designing industrial training material.

Coding Procedure: The analysis of the protocols to date has been qualitative and descriptive. We are still in the process of identifying the major components of the design problem space and arranging them in an explanatory fashion so as to build a model of the design process. We are not at a stage where we can engage in any quantitative or predictive analysis. But on the other hand, we are not limited to noting and relating everything we see. We have a rather explicit and constrained agenda: We want to know how the identified aspects of the DTE impact the DPS.

3.1. Extensive Problem Structuring

As noted earlier, many degrees of freedom exist in a design problem statement (or to put it in Reitman's terms, there is a lack of information). This lack of information impedes the creation of a problem space. Problem structuring is the process of finding the missing information and using it to construct the problem space (Simon, 1973a). It is the first step in any design activity. Large projects may require an alternation between problem-structuring and problem-solving phases. While some structuring is required in all problem situations, one of the hallmarks of design problems is that they require extensive structuring. The extent to which problem structuring is necessary and successful determines the nature and extent of the problem solving that will occur.

Each of the subjects in our experiment began by articulating and fleshing out their respective problem statements. This process proceeded through the follwoing steps: (1) gathering information from the design brief; (2) soliciting information and clarification from the experimenter through questions; (3) applying knowledge of legislative constraints (e.g., building codes, in-house company standards); (4) applying knowledge of "technical" constraints (e.g., "laws" of structural soundness, "laws" of learning); (5) attending to pragmatic constraints (e.g., time, money, resources at hand); (6) bringing to bear self-imposed constraints or personal knowledge; and (7) negotiating constraints. While each of these can bear considerable discussion, only the latter two will be addressed here. With respect to (6) two questions are raised: (i) what is the form and structure of this personal knowledge, and (ii) how and when is it brought to bear on the construction of the problem space? While we have no definitive answers to these questions, we do offer some preliminary observations. In the case of (7) we illustrate the process of negotiation and comment on when and why it might occur.

3.1.1. Form and Organization of Personal Knowledge

The personal knowledge our subjects used to construct their problem spaces was organized in rich, intricate chunks or schemas. Two types were discernible: *general schemas* and *domain specific schemas*. Generally, neither surface explicitly in protocols[x] but both are easily inferred from the

[x] Unless the subject stops to explain or rationalize, as one of our architects frequently did. Here is a typical excerpt from him: "Now, every building fitting into a site should be harmonious with that site. Nobody argues with that. The next thing, and compatible with the other buildings. Ah. We are going from a very antisocial period, where buildings were very antisocial and withdrawn, and aggressive, and impolite, such as the one we are standing in, to, ah, buildings which are pleasant, outgoing, gentle, ah, sophisticated and cultured ..."

situation-specific statements that the subjects make.

General schemas contain knowledge about the way(s) the world is. They are acquired over the course of a lifetime and are our primary means of dealing with the world. They consist of at least *procedural knowledge*, *abstract conceptual knowledge*, and knowledge of thousands of *patterns* (pictorial, linguistic, musical, etc.). Procedural knowledge is not open to introspection (Anderson, 1982) and thus does not surface in the protocols. However, both the abstract conceptual knowledge and some of the patterns are visible.

Abstract conceptual knowledge is the generalized knowledge—principles, laws, heuristics—which we extract and carry away from the totality of our worldly experience. While there is much structure and coherency in the organization of this knowledge, it does not necessarily constitute a theory. It is perhaps better characterized as knowledge fragments or "knowledge in pieces" (diSessa, 1985). It is instantiated and discernible in the problem space as *situation-specific* conceptual knowledge. For example, here is an excerpt from Subject-A's protocol:

(PF1) S-A: You, after all, you probably have your parcel or your precious letter and you want to get it out, stamp it, or ah, have a dialogue with a machine and see what, how much you have to pay. Your probably have to take it out from your bag, or whatever. So you do need a sort of protection.... I don't want them to get wet....

Underlying this verbalization are two knowledge fragments at the abstract, conceptual level—beliefs about the use of post offices and beliefs about when and where people do and do not like to get wet.

Knowledge of patterns is knowledge that is stored in such a direct way so that much of the original pattern or form is preserved (i.e., there is little generalization or abstraction). This may be voluntary, such as when students of poetry memorize lines of text or when architecture students draw and commit to memory the forms of specific buildings, or it may be involuntary, as in the case of a stimuli which the cognitive system is unable to fully comprehend and generalize. Instances of specific patterns are visible in all the protocols. Subject-A for instance, in attempting to reason about an automated postal interface, immediately retrieved and repeatedly used the "image" of an automated bank teller machine. But it was not some general conception of an automated bank teller but the specific Bank of America Versateller on Telegraph Avenue which he regularly uses.

(PF2) S-A: I don't want to have one booth after the other and having the lines, ah, like it were a Versateller, ah, kind of a service. Bank of America has that kind of approach, here on Telegraph. You have two, two Versatellers and usually have this long lines on the, ah, walk path. And who ever, ah, leaves first in one of the two, ah, then. So you have one single line for two machines. I am trying to avoid that....

Domain-specific schemas are built on top of the general schemas. They constitute the knowledge acquired during the years of professional training. They also consist of procedures, abstract conceptual knowledge, and patterns. Again, the procedures are not visible in the protocol. The abstract conceptual knowledge here seems to be less fragmentary and more "theoretical"[9] than in the case of the

[9] By 'theoretical' is meant only that it is more elaborate, complete, consistent, and organized.

general schemas. (This is not surprising considering that it was acquired as an organized, systematic body of knowledge.) For example, Subject-A has an elaborate mini-theory about the use and organization of space between buildings. His first sentences on viewing the site are:

(PF3) S-A: Well, what comes to my mind immediately, as I told you before when I was waiting [for] you, I was looking at, how this is set by pathways, this, this open space in between the sports court yard and these three buildings. And in thinking about the missed opportunity that people had here, of having a sort of more relaxed plaza, instead of being just a cross between these two directions. Which makes it very efficient, ah, but for sure it didn't, ah, give any contribution to the urban open space....

Similarly, Subject-I has a mini-theory about motivating, teaching, and imparting knowledge.

(PF4) S-I: The first thing we want to do with these people is try and sell them on a system. Any time you change somebody from an old system to a new system, or from what they are doing to what they're going to be doing, or what you're expecting them to be doing, you've got to give them a good positive reason. Why do I really? What's in it for me, you know.... This is positive reinforcement....

Several of these protocol fragments (PF1, PF2, PF3) are also examples of what we call *scenario immersion*. Scenarios are frequently occurring episodes in which designers recall and immerse themselves in rich intricate images from their past experience. The experience in question could have been acquired directly or vicariously through some symbolic medium (e.g., reading, watching TV). These episodes seem to play an absolutely crucial role in the process of generation and evaluation. For instance, the scenario in PF1 is used to generate the functional requirement "protection from rain." In PF2, the scenario is used to evaluate a proposed spatial configuration of APTMs. We say more about scenario immersion in section 3.2.

3.1.2. Application of Personal Knowledge

Personal knowledge structures and procedures are stored in long-term memory (LTM). Their indexing and retrieval is not well understood. Problem structuring is the process of finding and retrieving "relevant" schemta and instantiating them into the problem space. By *instantiation* is meant nothing more than the process by which a proposition of the content "All public buildings are required to have ramp access for the handicapped" is transformed into the proposition that "This building requires a ramp access." As it is construed here, problem structuring is not itself a problem-solving activity. But the extent to which it is successful does determine the amount of problem solving that will need to occur.

Subject-I was able to find, retrieve and instantiate a single powerful schema for designing training programs. The template came with slots marked for lessons, sections, subsections, etc. He merely had to fill in the blanks with the content of the particular course. He generates the required content by (i) asking the client (experimenter) for a list of tasks the secretary would be required to perform; (ii) drawing upon his own personal knowledge of Viewpoint; and (iii) consulting the Viewpoint manuals.[10]

[10] Some of the instructional design subjects actually used the Viewpoint manual to structure the task.

Finally, the selection of content is guided by an idealized cognitive model (ICM) (Lakoff, 1987) of what a secretary is: for example:

(PF5) S-I: All this isn't going to stay in this create and edit documents [lesson]. This is just looking at what's available, and what we are going to have to do. Because within this table of contents we've got related information—hardware requirements and so forth that has nothing to do with the secretaries, and foundation and environment. Secretaries couldn't care less.... And the logoff sheet properties. I, I wouldn't even teach the secretaries. That's none of *their business. They have no need for that information.* That I would teach your systems administrator....

Subject-A on the other hand seemed to find his design problem more of a challenge and exhibited somewhat different behavior. His initial structuring process took twenty minutes and resembled a "brainstorming" session. If the protocol for this phase is recorded as a directed graph, with the nodes forming individual "ideas" as they are uttered in temporal sequence, and the arcs connecting related nodes, then the result is a lattice structure. The density and distribution of the links suggest that there are really four smaller structures. First there are some site related constraints:

(PF6) S-A: You plotted those trees and that would really be a sin to touch them, I think. At least, the evergreens.... As far as seating space goes, the one just below the evergreens, I wouldn't touch all that corner....

Second there is a kernel idea:[11]

(PF7) S-A: *And what I thought is I shouldn't necessarily think of an enclosed building.* Cause, I am in the middle of an open space. It would be a contradiction to place a formal building there.

Third there are some ideas about the integration of site and structure:

(PF8) S-A: since this is the view towards the sports field, things happen over there after 5:00 p.m. I have seen people playing softball and ah, frisbee, and a lot of spectacular kind of activities. And I might take the opportunity of using this. So that people can be out there looking at the field. The sunset is going to be, ah, watched. Ah, my guess is that it would be a good opportunity to use it. And then now that I think of it, I am saying, well, I could even, ah, sort of think of something, some structure that might use the roof of my post office to be on a sort of more privileged position towards the field....

Lastly, there are some functional ideas about the flow of mail.

(PF9) S-A: I have to be concerned about the pick up service.... Ah, I need to be able to service the machine from behind and to have enough space to do so....

Thus, he was unable to retrieve a single unified plan or schema to guide his subsequent problem solving. He had to start his problem solving with at least four schemas and integrate them as he proceeded. This is a much more challenging situation than the one encountered by Subject-I.

Sometimes the domain-specific knowledge of the designer is insufficient to structure the problem. In such a case he first tries to use his general world knowledge; if this fails the problem may be avoided, abandoned or not even recognized. For example, the architect (Subject-A) had no

[11] The early generation and faithful development of a kernel idea is an intriguing phenomenon. It has been reported by several researchers, including Kant and Newell (1984) and Ullman et al. (1986). We do not have the space to pursue it here.

experience in designing user-transaction interfaces, but he was explicitly requested to do so in the design brief. He chose to assume a "Versateller type interface." When pressed by the experimenter to provide further details, he gave the following "explanation" for avoidance:

(PF10) S-A: the philosophy of it is that I hate an interface which is not human.... Let's leave it open. It might be through a keyboard, through a menu where you have a multiple selection and you have a ah, sort of Versateller mode to answer....

3.1.3. Negotiation of Problem Space Boundaries

Constraints as they occur are not always desirable. Negotiation of problem space boundaries is an interesting resultant phenomenon exhibited by most of our subjects. It is an attempt to shift problem space boundaries. Often it is done to minimize search effort by transforming the problem to fit an existing plan or template. This seems to be the motivation behind Subject-I's attempt. Subject-I, based on past experience, believes that training programs need some minimal instruction interaction. The instruction he was requested to design on this occasion was to be completely self-contained (i.e., no instructor interaction). He attempted to make the current task conform to his normal mode of operation:

(PF11) S-I: Ok. We can't negotiate you, ah, considering bringing these people in, ah, in possibly two groups of five, after hours, paid overtime or something, or is this already....

Sometimes negotiation is also used to enlarge and complicate the problem. Subject-A attempts to do this. On viewing the small triangular site he has been given for the proposed post office, he is not content to just build a post office but wants to redesign the whole area.[12]

(PF12)

S-A: So, given the fact we have that triangle [i.e., the site for the post office] over there as a limit. And I cannot exceed that I suppose?

E: Right, that, that....

S-A: I have to take that for granted?

E: I, I would think so.

S-A: That's the boundary of. You do not allow me to, to exceed in, in my area of intervention?

E: No, I think you should restrict it to that.

S-A: So, I am constrained to it and there is no way I can take a more radical attitude. Say, well, look, you are giving me this, but I actually, I, I'd come back to the client and say well look, I really think that you should restructure actually the whole space, in between the building. I'd definitely do that, if that was the case. You come to me as a client, and come to me with a triangle alone, I will give you an answer back proposing the whole space. Because, I, I think the whole space should be constructed. So, that there is an opportunity to finally to plan and that space through those, ah, this building, open up Anthropology and, and plan the three buildings together. So, as

[12] The subject is standing on a 9th floor balcony and has a bird's-eye view of the site.

to really make ah, this ah, a more communal facility....

The motive here is more difficult to speculate about. It could be a belief that this will result in a more effective artifact; a desire for a larger fee; exuberance and enthusiasm for rebuilding the world in one's own image; etc.

3.2. Extensive Performance Modeling

Four important aspects of the DTE converge to necessitate extensive performance modeling of the artifact (in its intended environment) in the design problem space.

1) Penalty for being wrong: It is a fact about the world that every action occurs in real time, consumes real resources, and has real consequences. In other words, it is impossible to set the world back as it was before the action. At best one can only take additional action (at additional cost) to remedy the situation, but traces of the original action will invariably remain. This is as true of bending one's little finger, uttering a sentence, walking to the grocery store, building a house or a freeway, or putting a man on the moon. The difference in each of these cases is in the cost and residue—the penalty for error. As the penalty for error increases, we respond by thinking through and anticipating as many consequences of an action as possible—*before acting*.

2) Autonomy of artifact: The artifact has an independent existence from the designer and must "make it on its own." The designer cannot be there to explain its significance or perform its function. For example, in the case of the stand-alone instruction, the instructional designer will not be in the classroom to respond to difficulties and questions of comprehension. He must anticipate the necessary interaction and respond to it in the structure of the artifact. Such anticipation/prediction requires extensive models of the artifact interacting in its intended environment.

3) Delayed/Limited feedback from world: Feedback from the environment is a major mechanism used by adaptive systems to enhance goal achievement in the face of variable environmental factors. One of the most dramatic consequences of the structure of the DTE is that the feedback loop is delayed. The design is being developed between time t and $t+1$ (see Fig. 1) but it does not interact with the world until time $t+3$. But this for all practical purposes is a point of no return. Resources have been expended and the damage has been done. The feedback from this point can not guide the designer in the current project, but only the next "similar" project. To guide the current problem solving the designer must simulate or generate his own feedback between times t and $t-1$.

4) Temporal separation of specification and delivery: There is a linear, temporal separation between artifact specification and delivery. In Fig. 1 the specification is complete at time $t+1$ and the artifact constructed in the world at time $t+3$. Ideally the artifact is completely specified before construction begins.[13] This temporal separation enables the designer to

[13] This is of course not always the case. Fast-tracking is a case of substantial parallel processing. But, even here, there are significant self-contained modules—and errors are expensive.

model artifact performance—in the problem space or some external medium—to minimize damage and the expenditure of more substantive resources.

Performance modeling is necessitated by the first three aspects and enabled by the fourth.

Modeling is both internal and external to the problem space. Some of the possibilities, and the sequence in which they are used, are as follows: (i) entailments of designer's ICMs, (ii) scenario immersion, (iii) pictorial models, (iv) mathematical models, (v) mock ups, (vi) surveys, (vii) computer simulations, etc. Our subjects did not have the time or resources to make use of all these modeling devices—though they all pointed out when they would normally use them. They were basically restricted to their problem space and paper and pencil. This meant that they could take advantage of only the first four types of models. We will restrict our discussion to a few comments about the first two.

The designer's ICM of the world allows for quick and automatic inferences. We have already encountered an example in PF5 where Subject-I uses his "secretary ICM" to quickly evaluate whether to include certain material in the lessons. Such inferences do not seem to require any effort. They fall out automatically from the designer's idealized cognitive model of the world.

Scenario immersion is a more elaborate process whereby the designer pulls out a relatively concrete scenario from his past experience and immerses himself in it. Knowing how the scenario actually transpired, he draws upon similarities between the scenario and the current situation to calculate the entailments of the current situation. It is a strategy of both the first and the last resort. For example, we saw in PF2 how Subject-A evaluated a one possible spatial configuration of APTM machines by doing a mapping between it and a previously encountered similar situation, the consequences of which he has had first-hand experience. Subject-M, in determining the size and height of APTM machines wanted to do a formal study to see how people would use the machine. However, (perhaps knowing a formal study is not possible in the circumstances), he immediately and without prompting indulges in scenario immersion.

(PF13) S-M: Ok. I think [we need...] user group studies about how.... they would do the transaction. I think there is something about how...they're going to use it. Maybe, most student maybe riding bikes sometimes. Or most people, we expect them to walk, walk in. But sometimes maybe students [are] kind of lazy, or maybe they ride their bike or moped....

While their external models varied according to task demands and their pre-existing notational systems, the scenario immersion strategy was common across all subjects.[14]

3.3. Personalized/Institutionalized Evaluation Functions and Stopping Rules

It has been noted by many people (e.g., Rittel and Webber, 1974) that there are no right or wrong answers in design situations, but only better and worse ones. This has two interesting consequences at the level of the problem space. First, it means that evaluation functions are often

[14] Not only does the scenario immersion phenomenon play a crucial role in performance modeling, it also seems to be instrumental in generation. However, we do not discuss this aspect of it here.

personalized or at least institutionalized.[15] This is quite apparent in the above uses of ICMs and scenario immersion. Second, the point at which a design is complete is a function of cognitive and personal resources. Subject-I asked to stop because he was tired. Subject-M reported he could not proceed any further without doing a mock-up of the APTM. As we did not have the resources there for him to do so, he used this as a reason to terminate the session.

3.4. Limited Commitment Mode Control Strategy with Nested Evaluation Cycles

In section 3.2 we discussed the importance of performance modeling. Ultimately the purpose and value of this is to enable the designer to anticipate the performance of the artifact and the consequences of releasing it in the world. Since what matters is the performance of the final, complete artifact (at time $t-3$), one possible strategy is to delay evaluation until the specification is complete (at the end of time $t-1$). Evaluation at this point would certainly yield as good a value as possible, short of direct feedback at time $t-3$. But given the time, cost, and complexity involved in the design phase itself, it is neither optimal or feasible. That is, quite apart from the time and costs involved in generating a complete design and then having to scrap it and start all over again, it is a fact about adaptive systems that they require continual feedback when engaged in any goal seeking endevour. It is simply not possible for people to work for months on end without having any indication as to the value and status of the work with respect to the goal. So not surprisingly, we found that our subjects did not wait until the artifact was completely specified to evaluate its performance.

Since the design unfolds in a quasi-linear sequence, generally starting with a kernel idea that is transformed and augmented until the final form emerges, another possible strategy is to evaluate components of the artifact as they are being generated. This would result is a linear sequence of short generate-evaluate cycles. While this is cognitively a very tractable strategy it can arrest design development by requiring strict adherence to earlier decisions. That is, a decision made at one point, while attractive in that local context, may be inappropriate in a later, more complete context. With this strategy one would be stuck with the earlier decision. Our subjects did not use this control strategy either.

Instead, all our subjects used a *limited commitment mode control strategy* (LCMCS) which incorporates the best of both worlds. It is cognitively tractable, enhances design development, and gives good evaluation results. It is necessitated by the essentially sequential nature of symbolic processing and made possible by the fact that the design phase is separate from, and prior to, the delivery phase.

If one looks at the design process at any given time, one finds that there are at least three contexts that the designer needs to attend to: (i) the component of the artifact currently being generated or focused on; (ii) the complete artifact in its current state (i.e., the design so far); and (iii) the projection of the artifact in its complete state (i.e., the final design). The LCMCS allows the designer to take each of these contexts into consideration.

---

[15] By 'institutionalized' is meant accepted by a group or organization with which the designer associates himself. For example, in the case of Subject-I, this means in-house company standards and practices. In the case of the architect, it might be some "movement" such as Bauhaus, Postmodernism, etc.

As a first option, the designer can evaluate a generated or focused component on its own and make a decision to accept or reject it. For example, the instructional designer thought of including the component "start with basics and finish with more complex" in a subsection entitled "What will be Trained." He rejects it even before verbalizing it. (It surfaces only when the experimenter intervenes with his question.)

(PF14)

S-I: Ok, we've overviewed the course now just as far as the selling features. Now we're going to do a little bit of overview of what to expect. [writing: "What Will be Trained"] Ah, now what we will train. Ok, and we put that over.... [writing: "Six 1-hour Sessions"] We're going to, oh hell, that's bullshit.

E: What was bullshit?

S-I: Start with basics and finish with more complex. Well of course. What in the hell else would you be doing? I am not going to step you right off the end of the Titanic and ask you to swim....

What matters for present purposes is that the evaluation of the component was not done in the context of the design but strictly locally, on its own terms.

Second, the designer can evaluate a generated or focused component in the current context (i.e., the context of the design so far). This practice results in a better evaluation function and an increase in the number of options. He can choose to reject or accept the current component or he can choose to reject or modify some previous decision to make the current one acceptable. For example, at one point, Subject-I makes a decision to the effect that secretaries don't need to know about "waste baskets" (an icon used to delete computer files). A little further down he decides that they should know how to recover deleted icons. Then he realizes that the only way they can do this is if they know how to use "waste baskets." At this point he can simply reject the later decision of teaching the secretaries about recovering deleted icons, but instead he decides to undo the previous decision and include a section on "waste baskets." This now makes it possible to stick to the second decision of teaching about the recovery of deleted icons.

Finally, the designer can evaluate the generated or focused component in a later more complete context (at a later time), further increasing accuracy and options. In this situation, he can accept or reject the current component, as in the first case; modify some previous decision to make the current one acceptable, as in the second case; but in addition has the option to modify some future decision to make the current one acceptable. For example, Subject-A during his initial structuring phase had an idea for using the roof structure of the post office as a seating platform for viewing the sports field:

(PF15) S-A: I could even, ah, sort of think of something, some structure that might use the roof of my post office to be on a sort of on more privileged position towards the field....

But later when he calculated the size of the structure and realized how small it would be (i.e., reevaluated it in the current, more complete context), he abandoned the earlier idea:

(PF16) S-A: The thought that I had before, that I might use, the envelope itself, the form, the roof, ah, the walls, to, to implement some sort of, ah, landscape element, so as to have a major view towards the sports field. That I am denying now.... I really am coming back

> to this and seeing that, after all, I won't have huge lines. After all I just have 3 booths and a roof. That's what I really have here. So, I'm sort of seeing the extent, ah, to which this problem will be heading to.

3.5. Making and Propagating Commitments

A design task is not complete until the artifact is completely specified. A specification is a complete, procedural, and declarative description, which when executed by an external agent results in the construction of the artifact. It is not sufficient to wave one's hands and talk about the artifact in some general terms. One must actually make, record, and propagate decisions as one proceeds, otherwise one will have nothing to show at the end of the session. Each of our subjects did explicitly record and propagate their decisions.

An interesting tension exists between the LCMCS and the need to make commitments—between not acting and acting rashly; between being Hamlet and being Laertes. Designers are adept at negotiating this tension between keeping options open for as long as possible and making commitments.

3.6. Solution Decomposition into Leaky Modules

A major cognitive strategy for dealing with large complex problems is through decomposition. Decomposition was a major step in the normative models of the design methodology movement (e.g., Alexander, 1964). It has since been questioned and discredited as overly simplistic and even harmful to the design process. As Alexander (1965) subsequently noted, "A city is not a tree; it is a semi-lattice." Or in Simon's (1962, 1973b, 1977) vocabulary, the world is only *nearly* decomposable. But what is to be made of the *nearly*? Some interpret it to mean that one can not talk about solution decomposition in any significant sense. Others assume it can be ignored and continue to do clean, tree-like decompositions (e.g., Brown and Chandrasekaran, 1985).

Our data shows extensive decomposition. Each of our subjects quickly and automatically decomposed their problem and developed their solution in a dozen or so modules. Subject-M's modules were things like key pad, screen, stamp dispensary, parcel depository, and weighing mechanism. The decompositions were discipline specific. They were not invented anew for the problem but seemed to be part of the designers' training and practices. However, equally important, the subjects did not treat the modules as strictly encapsulated but rather as *leaky modules*. A decision made in one module could have consequences in several others. The subjects seemed to have some sort of ongoing monitoring process that looked for interconnections across modules.

The subjects dealt with the problem of "leaks" in one of two ways. One method was to plug the leaks by making functional level assumptions about the interconnecting modules (see section 3.7). This method enabled them to bring closure or encapsulation to a module and make it cognitively tractable. For instance, in designing the first lesson, Subject-I did not have to attend to the details of the third lesson. It was sufficient to make some high-level functional assumptions about it. Similarly, in considering the height and angle of the APTM key pad, Subject-M did not attend to the details of the stamp

dispensary. A second method of dealing with "leaks" was to engage in opportunistic behavior—to actually put the current module on "hold" and to attend to some of the interconnecting modules right there and then.

3.7. Abstraction Hierarchies Mediate Transformation of Goals to Artifact

The input to the design process is generally a set of goals or intentions. The output of the process is generally a specification of an artifact. The goals come substantially from the client (though are elaborated in discussion with the designer) and are a statement of the behavior he wants the artifact to support. The artifact specifications are substantially generated by the designer (though the client's brief may provide some guidelines at the level of the artifact) and specify those aspects of the artifact that he considers to be causally relevant in the given circumstances. Conceptually or logically, it is tempting to say that the transformation from goals to artifact specifications is mediated by functional specifications (see Fig. 3). On this account one gets a story whereby the intentions are carried out *by means of* the functioning of the artifact, and the function is carried out *by means of* the causal structure of the artifact. Both function and causal structure have to fit the intentions, but they are only constrained, not determined, by them. In fact the intentions constrain (underdetermine) function, and function constrains (underdetermines) causal structure (see Fig. 3).

Fig. 3 approx here

Such explicit mediation is sometimes visible in our data. For example, Subject-M, when determining the components and configuration of the APTM, began with a very functional vocabulary:

(PF17) S-M: I think that functioning-wise we have some criteria. Ah, it's supposed to fulfill the requirement of user to purchase the stamps, mail the letters, and weigh parcels and mail it. Certainly there also will be register, should be something that can do the function for registering letters. And ah, certainly we expect it to be user-friendly and without requiring any training, and transparent to user....

At this point there is no indication of how these functions will be realized. A few minutes later they are mapped onto device components on a one-to-one basis:

(PF18) S-M: So I would assume there is input and output devices...and we got to also have depository...for letters and parcels, and something for...delivering device for stamps.... And we also need some device to weight....

But generally, the story that emerges from the data is not quite so clean and is closely connected to the near-decomposability phenomenon of the previous section. The functional specifications and the causal structure specifications are not two distinct ontological categories but the same category under different descriptions. Functional specifications treat the artifact, or some component of it, as a black box and attend only to the input and output. They basically answer the question "what function will this artifact, or this part of it, accomplish?" Artifact specifications detail the causally efficacious

structure of the artifact. They answer the question "how is the function to be accomplished?" For example, during the course of designing the first lesson in the training package, Subject-I worked with several different modules, interconnected in various ways. Some of these modules were: lessons, sections, subsections, paragraphs, sentences, and the choice and arrangement of lexical and grammatical elements. This corresponds to what we called a solution decomposition in the previous sub-section. In addressing each of these modules the designer may choose to do it at various levels of abstraction or detail. The functional-causal structure distinction is just a special case of this abstraction process.

The status of any module vis-à-vis the functional-causal structure distinction depends on whether a *what* or *how* question is asked of the module. For example:

- What is the function of this lesson?
- How is it going to achieve this function?
  (By means of these sections.)
- What is the function of these sections?
- How are they going to achieve their function?
  (By means of these subsections.)
- What is the function of these subsections?
- How are they going to achieve their function?
  (By means of these paragraphs.)
- etc.

In asking the different questions, the designer is choosing to attend to different levels of detail. Ultimately, this regress must bottom out at a level where the artifact is completely specified. There are some interesting observations to be made as to where it bottoms out and the number of levels a designer explicitly considers.

Our data indicates that the number of levels explicitly attended to by a designer is a function of his experience and familiarity with the task, availability of relevant knowledge, and personal preferences. The more routine a task is, the more quickly and directly the designer can get to the low-level details, if he so chooses. He knows by experience what type of artifact supports what type of goals and does not have to reason through it via "first principles." Of our three subjects, Subject-I found the task quite routine and traversed the abstraction hierarchy quite quickly. Subject-M, as noted earlier (PF17 & PF18), did cascade down several levels of function-artifact specifications. Subject-A, when confronted with designing the automated mail-handling system for the post office, dealt with it in strictly functional terms. He simply did not have the knowledge to specify lower-level details.

However, Subject-A consciously did something that was rather interesting. In determining the configuration and location of the post office building, he purposefully stayed at a highly abstract level for an extended period of time so as not to crystalize or commit himself too soon to low-level details:

(PF19) S-A: I am constantly referring to that sketch by the way. As you can see it's ah, although it's the lousiest of them all, it still, still something that I, I, I, and I am not willing to do any other sketch at the moment. Because I, I am really, trying to figure it out and I am doing it at an abstract level. So, that...the flow is not affected by the crystalization of an idea....

Thus training, personal preferences, style, and a number of pragmatic factors can affect the number of abstraction levels that are considered and how quickly one descends the hierarchy. This point is tied to the personalized evaluation function and stopping rules observation discussed in section 3.3. Descending too soon or not descending[16] at all is a common mistake of novice designers. This relates to the earlier point about the tension between the LCMCS and the making of commitments.

The level of detail at which the designer chooses to bottom out depends on professional conventions and standards, personal preferences, style, and a host of pragmatic factors. Subject-I, for example, did not stop at the specification of the actual words and sentences but went on to also specify page layout and typeface. But he did not have to stop there either. He could also have specified the chemical composition of the ink or the tensile strength of the paper. He chose not to. He left it as someone else's responsibility. He simply assumed that they would function in the "normal way"—that the ink would not dissolve and the paper would not fall apart—and did not feel the need to provide any specifications for them. Every design profession has some conventions in this respect, and there is always some freedom either way that the designer may exercise at his discretion.

3.8. Use of Artificial Symbol Systems

Designers often use artificial symbol systems to filter and focus information and augment memory and processing. These systems are so crucial for the problem-solving process that if they do not pre-exist they have to be invented before the design can proceed.[17] Their use and importance can be seen most dramatically in the case of architecture. It is possible to recognize at least seven different symbol systems (six of them artificial) in the architect's repertoire (see Fig. 4). Roughly they are (i) natural language, (ii) topology ("bubble diagrams"), (iii) similarity geometry (rough sketches), (iv) Euclidean geometry (plans, elevations, sections), (v) affine geometry (isometrics), (vi) projective geometry (perspectives), and (vii) models or mockups. (Admittedly, the correspondence between the formal geometries and the architect's various drawings is only approximate, but it does serve to highlight the richness and variety of artificial symbol systems that are actually used.)

Fig. 4 approx here

The symbol systems from topology to Euclidean geometry form a sort of a hierarchy. In fact, they map onto and support the abstraction hierarchy discussed in the previous section. It is possible to make and represent distinctions at the lower levels which the higher levels do not support. Similarly, it is possible to make and represent distinctions at the higher, more abstract levels, which can only be

[16] One of our instructional designer subjects stayed at a very high abstract level and refused to come down. The result was that he had no artifact specifications to show at the end of the period

[17] One of our subjects realized that he did not have an appropriate symbol system for the development and specification of the artifact and tried to develop one as he went along. The development of symbol systems can be seen on an institutional scale in the case of the emerging scripting and mazing systems for interactive videodisc

made at the lower levels in a hidden or obscure fashion. For example, metric distinctions are preserved in Euclidean geometry but not in topology, and while every proposition of topology is trivially true in Euclidean geometry, topology does not come into its own until one abstracts away from metric and other details.

Subject-A in his two-hour session used the symbol systems of natural language and similarity geometry. There are two interesting things to note in his use of these systems. (1) Moving between the systems *automatically* commits him to a level of detail by selectively highlighting and hiding information. (2) Within a single symbol system he constructs multiple representations of the artifact. In both cases we want to note that these external representations are not for communicating something after the fact. They serve an indispensable role in the generation, evaluation, and decision-making process. Once decisions are made, symbol systems serve to record and perpetuate them.

As an illustration of the first point, consider the following sequence of protocol fragments and the accompanying diagrams in which Subject-A determines the form and configuration of the post office building:

(PF20) S-A: But I could eventually have one single space, where all the, ah, mail is, is delivered. Which eventually would open up in a single way and have the booths orbiting around it. So that a given line might occur here, another one here, and another one there.... Now what I see is a more enclosed to itself structure. By that I want to say is that there is an inner core and then this roof extending around it....

Along with this verbalization was the concurrent realization of the geometric form in Fig. 5.

Fig. 5 approx here

The relationship between the verbalization and the diagram is a one-to-many mapping. The diagram contains several elements which the verbalization does not. It contains and makes very explicit information on the rough size (relative to users) and shape of each unit, the configuration of the units, and how the designer envisions the lines forming. This is not an accident. It is simply not possible to draw the artifact in similarity or Euclidean geometry without making commitments on these issues, whether you are ready to or not.[18] In fact, a few minutes later, while examining Fig. 5, Subject-A expresses surprise when he realizes the full extent of his commitment and commences to modify it.

(PF21) S-A: I don't want to, to affect the type of line that might happen. Why did I draw this, ah, like something that sticks out? Ah, no. I actually want to minimize even more. So, the way I see it now is I'll have to, ah, the booths [are], conceived, probably in such a way that, the element itself is, is really minimized as, as, ah, formal or volumetrical type of ah, intervention. We have a main structure and 1, 2, 3 interfaces, and the main axis.... This seems to work well....

---

[18] In actual similarity geometry, size, of course, is not preserved

Accompanying this verbalization is the diagram in Fig. 6. While substantially different from the diagram in Fig. 5, it is consistent with the original verbalization in PF20.

Fig. 6 approx here

Each sketch highlights information not explicit in the verbal description. As the information is explicated, it can be attended to in subsequent generate-evaluate cycles. Much of Subject-A's problem solving involves traversing abstraction levels via the corresponding symbol systems. Learning to traverse this hierarchy has some serious consequences for design development and crystalization. One must know when to use which system so as not to commit oneself too soon and thereby prematurely arrest design development. At the other extreme one must learn not to stay at the higher abstract levels for an overly extended period of time and thereby produce nothing. This observation is of course related to the earlier mentioned tension between the LCMCS and the necessity to make commitments.

To illustrate the second point, we note that Subject-A constructed four distinct representations of the artifact within the system of similarity geometry: site plans, floor plans, elevations, and sections. Furthermore, he attended to the various aspects of the building as they were being drawn: for example, he calculated the vertical dimensions of the structure when drawing the elevation (see Fig. 7), not when working on the plan:

(PF22) S-A: So, maybe, ah, I should go on to a section now and see how this is ah, happening, with more precise measures [meaning roof overhang and the glare on the monitors].... Ah, 6 feet I envisioned this to be very low anyway.... Probably 2.4 meters, 2.2 meters even.... So I'd say that 8 feet will be the maximum height.... Ah, probably we need about 2 or 3 feet to have all the equipment.... And the lower part of the display monitor and, and keyboard will be perhaps 3 feet, 3.5 feet perhaps from the ground level.

Fig. 7 approx here

---

4. CONCLUSION

This study has identified eight significant invariants in the design task environment and characterized their impact of the design problem space. Fig. 2 serves as a succinct summary of both our strategy and findings. To repeat, our major empirical findings are the following characteristics of the design problem space: (A) extensive problem structuring, (B) extensive performance modeling, (C) personalized/institutionalized evaluation functions and stopping rules, (D) a limited commitment mode control strategy, (E) the making and propagating of commitments, (F) solution decomposition into "leaky modules," (G) the role of abstractions in the transformation of goals to artifact specifications, and (H) the use of artificial symbol systems. But in addition to noting these features, we also made

"explanatory connections" between them and the invariant features of the design task environment. We make no claim for completeness and fully expect our characterization to grow and evolve as we examine more of our data. But we do expect our strategy of viewing design as a radial category, taking the design task environment seriously, and examining data from several design disciplines to be of continuing value in the future. At this stage, we cautiously suggest that while singularly these features may be found in non-design problem spaces, collectively they are the invariant hallmarks of the design problem space. We now conclude the paper by indicating some implications for CAD systems, noting some methodological shortcomings and suggesting directions for future research.

4.1. Implications for Computer-Aided Design Systems

Typically, CAD systems provide designers with a variety of tools for modeling the anticipated performance of an artifact during the design process. Our characterization of generic design and our empirical observations suggest that there are several ways in which such systems could be enhanced.

We noted that design characteristically involves problems with many degrees of freedom, requiring substantial collection of information, problem structuring, and negotiation. Much of this information comes from external sources or the prior experience of the designer. At first blush, hypertext tools would seem to be appropriate for such activities. However, as noted by Halasz (1988), making hypertext systems that permit cheap input and restructuring is still a major research issue.

Design inherently involves the use of design abstractions, nested generate and evaluate cycles, and a limited-commitment mode control strategy. This suggests that designers should be able to inexpensively specify design abstractions and evaluate designs at any level of abstraction. The CLU language for software design is an attempt along these lines (Liskov and Guttag, 1986). It is essentially a variant of an object-oriented programming language that allows software designers to develop procedural and data abstractions and specify the preconditions and entailments of these abstractions without immediate concern for their implementation. The fact that designers appear to mix formalisms in their representations of artifacts suggests that we have substantial work to do in this area.

Representation is an important issue in itself. First generation CAD systems viewed the designer's notes and drawings only as communicative devices. Our studies confirm the findings of Ballay et al. (1984) and Ullman et al. (1986) that this is simply not the case. The designer's notes and drawings play a crucial role in design development by selectively focusing and filtering information and augmenting memory and processing. This speaks for the need to develop computational environments which can support a wide range of symbol systems.

Finally, we should remark on the potential role of AI in CAD systems. AI is especially appropriate for propagating the entailments of closed-world models, as is typically done in theorem-proving programs or problem solving programs that deal with well-structured problems. It does not fare as well in tasks with changing world models; ones that are continually influenced by knowledge brought in from the external world or from past experience. This would seem to imply that we should not expect AI to provide highly automated design systems for anything but the most routine and well-

structured problems that arise during design. However, research on hierarchical planning could provide tools for representing and evaluating abstract design plans. Research in knowledge acquisition tools could influence the development of CAD systems that acquire new design abstractions and evaluations. Research in case-based retrieval and reasoning could provide tools to augment designers' use of prior knowledge in design. Intelligent advice or help systems that use knowledge of particular design tasks, and on-line "pattern books" might be particularly useful aids for novice designers or as warehouses for the design knowledge in particular disciplines or institutions.

4.2. Principle Shortcomings and Limitations

As the work currently stands, there are three principle shortcomings. The first is that the whole analysis is based substantially on three protocols, one each from three of the many design disciplines. In the short term we justify our experiment design by noting that the methodology used is qualitative rather than quantitative. It does not require large numbers of subjects. As has been argued by Anzai and Simon (1979) there is much to be gained by the detailed analysis of a single protocol. Over the long term, we recognize the shortcoming and are continuing to analyze additional protocols.

The second shortcoming is that we have not used a formal procedure for coding the protocols. Neither has there has been any independent coding of the protocols. Again, over the long term, we recognize this shortcoming as serious. In the short term, we note that the categories and conclusions were arrived at through much argumentation and compromise with colleagues with first-hand knowledge of the data.

The third shortcoming is that only design problem protocols have been examined. This only allow us to make the weak claim that we have identified certain invariants in the design problem space. It does not permit the additional claim that these invariants are not (collectively) found in nondesign problem spaces. This latter claim is desirable for the motivation of generic design as a useful theoretical construct. But it requires the examination of nondesign protocols. The comparison of nondesign problem spaces with design problem spaces is a matter of ongoing concern.

4.3. Future Work

This investigation has been a first-pass, breadth-first look at design problem solving. We have tried to lay out the major pieces of the design problem space and explain or justify them by an appeal to the design task environment and the structure of the IPS. A logical extension of this work would be to push the analysis further and to derive a process model of design from it.

In concentrating on the big picture, we have had to resist the temptation to delve deeply into any single feature of the problem space. Of particular interest to us are the phenomena of scenario immersion, leaky modules, and the use of artificial symbol systems. Each of these promises to be a rich and intricate field of study.

Also, we have not said anything about the differences in the problem spaces of our three subjects. We have noticed some interesting differences in their knowledge bases, the external symbol

systems they use, and their cultural and professional values and practices. However, any conclusions in this regard must wait until we gather and analyize additional data.

References

Akin, Omer. "An Exploration of the Design Process," *Design Methods and Theories*, vol. 13, no. 3/4, pp. 115-119, 1979.

Akin, Omer. *Psychology of Architectural Design*, Pion Limited, London, 1986.

Alexander, Christopher. *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA, 1964.

Alexander, Christopher. "A City is not a Tree," *Architectural Forum*, vol. 122, pp. 58-62, April-May, 1965.

Anderson, John R., "Acquisition of Cognitive Skill," *Psychological Review*, vol. 89, no. 4, pp. 369-406, 1982.

Anzai, Yuichiro and Simon, Herbert A., "The Theory of Learning by Doing," *Psychological Review*, vol. 86, no. 2, pp. 124-140, 1979.

Archer, L. Bruce, "The Structure of the Design Process," in *Design Methods in Architecture*, ed. G. Broadbent & A. Ward, George Wittenborn Inc., N.Y., 1969.

Ballay, Joseph M., Graham, Karen, Hayes, John R., and Fallside, David, "CMU/IBM Usability Study: Final Report," Communications Design Center Technical Report No.11, Carnegie-Mellon University, 1984.

Brown, David C. and Chandrasekaran, B., "Knowledge and Control for Design Problem Solving," Tech. Report July 1985, Laboratory for Artificial Intelligence Research, Dept. of Computer and Information Science, The Ohio State University, 1985.

Cross, Nigel, "Understanding Design: The Lessons of Design Methodology," *Design Methods and Theories*, vol. 20, no. 2, pp. 409-438, 1986.

Eastman, Charles M., "On the Analysis of Intuitive Design Processes," in *Emerging Techniques in Environmental Design and Planning*, ed. G. Moore, MIT Press, Cambridge, MA, 1969.

Eastman, Charles M., "Recent Developments in Representation in the Science of Design," *ACM & IEEE 18th Design Automation Conference*, pp. 13-21, 1981.

Ericsson, K. Anders and Simon, Herbert A., *Protocol Analysis: Verbal Reports as Data*, The MIT Press, Cambridge, Massachusetts, 1984.

Greeno, James G., "Natures of Problem-Solving Abilities," in *Handbook of Learning and Cognitive Processes, Volume 5: Human Information Processing*, ed. W. K. Estes, Lawrence Erlbaum Associates, Hillsdale, N.J., 1978.

Halasz, F. G., "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems," *Communications of the ACM*, vol. 31, pp. 836-852, 1988.

Hayes, John R. and Simon, Herbert A., "Understanding Written Problem Instructions," in *Knowledge and Cognition*, ed. L. W. Gregg, Lawrence Erlbaum Associates, Potomac, Maryland, 1974.

Jeffries, Robin, Turner, Althea A., Polson, Peter G., and Atwood, Michael E., "The Processes Involved in Designing Software," in *Cognitive Skills and their Acquisition*, ed. J. R. Anderson, Lawrence Erlbaum, Hillsdale, N.J., 1981.

Kant, Elaine and Newell, Allen, "Problem Solving Techniques for the Design of Algorithms," *Information Processing and Management*, vol. 20, no. 1-2, pp. 97-118, 1984.

Kant, Elaine, "Understanding and Automating Algorithm Design," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 11, pp. 1361-1374, 1985.

Lakoff, George, *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*, The University of Chicago Press, Chicago, 1987.

Lawson, Bryan R., "Cognitive Strategies in Architectural Design," *Ergonomics*, vol. 22, no. 1, pp. 59-68, 1979.

Liskov, B. and Guttag, J., *Abstraction and Specification in Program Development*, MIT Press, Cambridge, MA, 1986.

Newell, Allen and Simon, Herbert A., *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, N.J., 1972.

Newell, Allen, "Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category," in *Attention and Performance VIII*, ed. R. S. Nickerson, Lawrence Erlbaum, Hillsdale, N.J., 1980.

Perkins, D. N., *Knowledge as Design*, Lawrence Erlbaum Associates, Publishers, Hillsdale, N.J., 1986.

Reitman, Walter R., "Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-Defined Problems," in *Human Judgements and Optimality*, ed. M. W. Shelly II & G. L. Bryan, John Wiley and Sons, N.Y., 1964.

Rittel, Horst W. J. and Webber, Melvin M., "Dilemmas in a General Theory of Planning," *DMG-DRS Journal*, vol. 8, no. 1, pp. 31-39, 1974.

diSessa, Andrea A., "Knowledge in Pieces," *Constructivism in the Computer Age*, pp. 1-24, 1985. (Fifteenth Annual Symposium of the Jean Piaget Society, Philadelphia, June 1985.)

Simon, Herbert A., "The Architecture of Complexity," *Proceedings of the American Philosophical Society*, vol. 106, pp. 467-482, 1962.

Simon, Herbert A., "The Structure of Ill-Structured Problems," *Artificial Intelligence*, vol. 4, pp. 181-201, 1973a.

Simon, Herbert A., "The Organization of Complex Systems," in *Hierarchy Theory*, ed. H. H. Pattee, G. Brazileer, N.Y., 1973b.

Simon, Herbert A., "How Complex are Complex Systems?," *Proceedings of the 1976 Biennial Meeting of the Philosophy of Science Association*, vol. 2, pp. 507-522, 1977.

Simon, Herbert A., *The Sciences of the Artificial (Second Edition)*, MIT Press, Cambridge, MA, 1981.

Steier, David M. and Kant, Elaine, "The Role of Execution and Analysis in Algorithm Design," *IEEE Transactions on Software Engineering*, vol. SE-11, no. 11, pp. 1375-1386, 1985.

Thomas, John C. Jr., "A Design-Interpretation Analysis of Natural English with Applications to Man-Computer Interaction," *International Journal of Man-Machine Studies*, vol. 10, pp. 651-668, 1978.

Thomas, John C. and Carroll, John M., "The Psychological Study of Design," *Design Studies*, vol. 1, no. 1, pp. 5-11, 1979.

Ullman, David G., Stauffer, Larry A., and Dietterich, Thomas G., "Preliminary Results of an Experimental Study of the Mechanical Design Process," Tech Report 86-30-9, Dept. of Computer Science, Oregon State University, 1986.

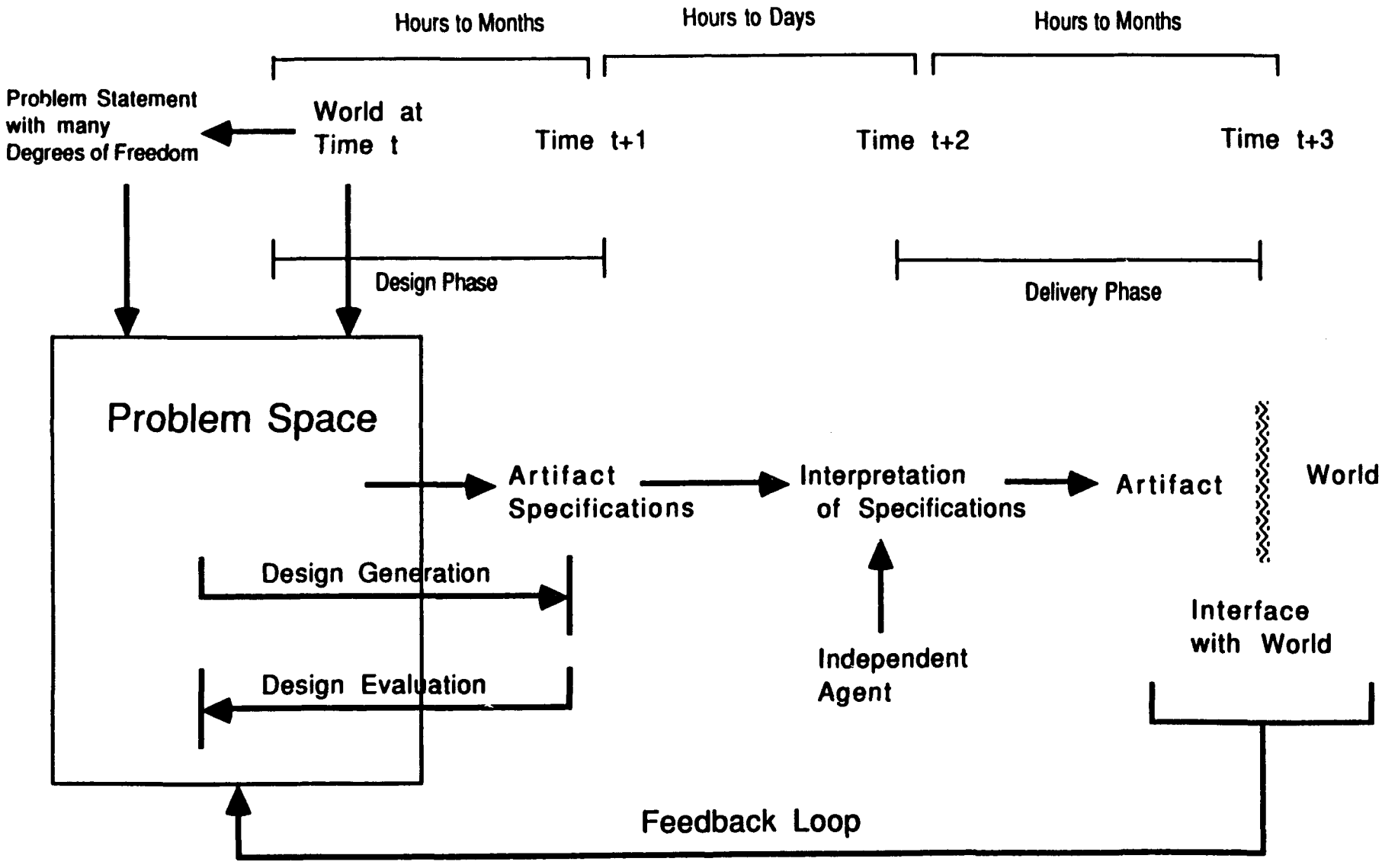


Fig. 1: Structure of a Prototypical Design Task Environment

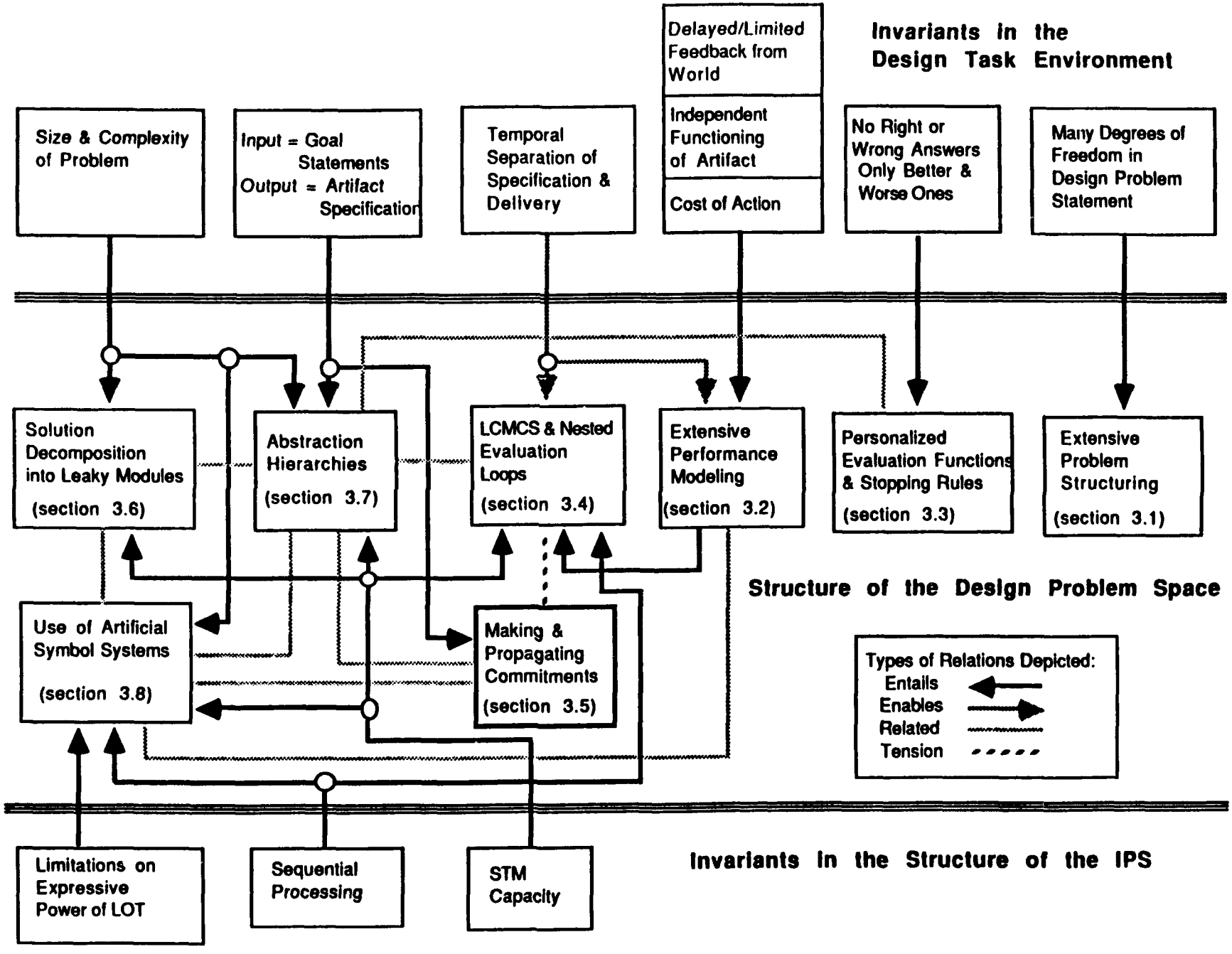


Fig. 2: The Design Problem Space, as Structured by the DTE and the IPS. See text.

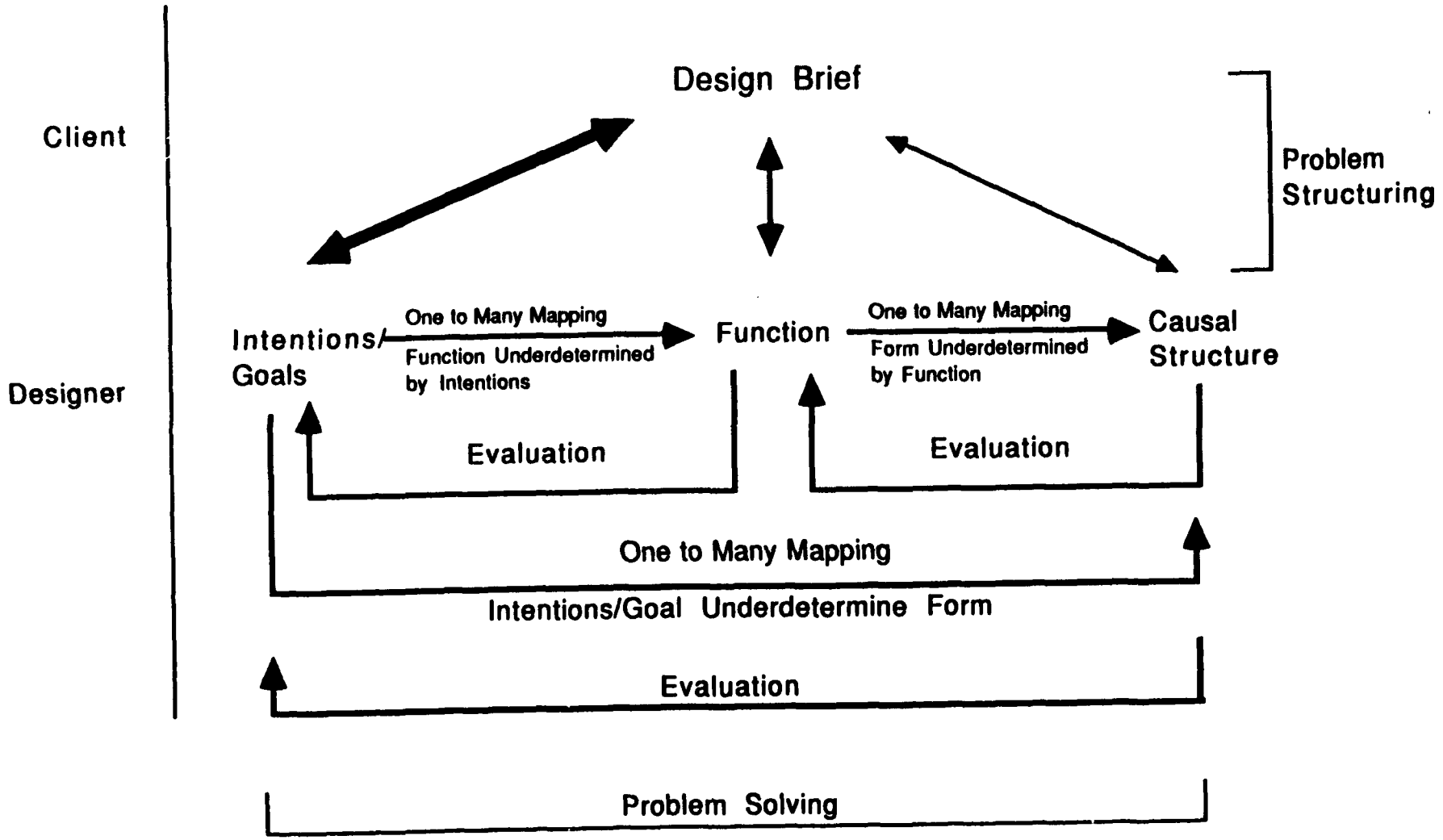


Fig. 3: Conceptual or Logical Structure of the Transformation of Goals to Artifact Specifications

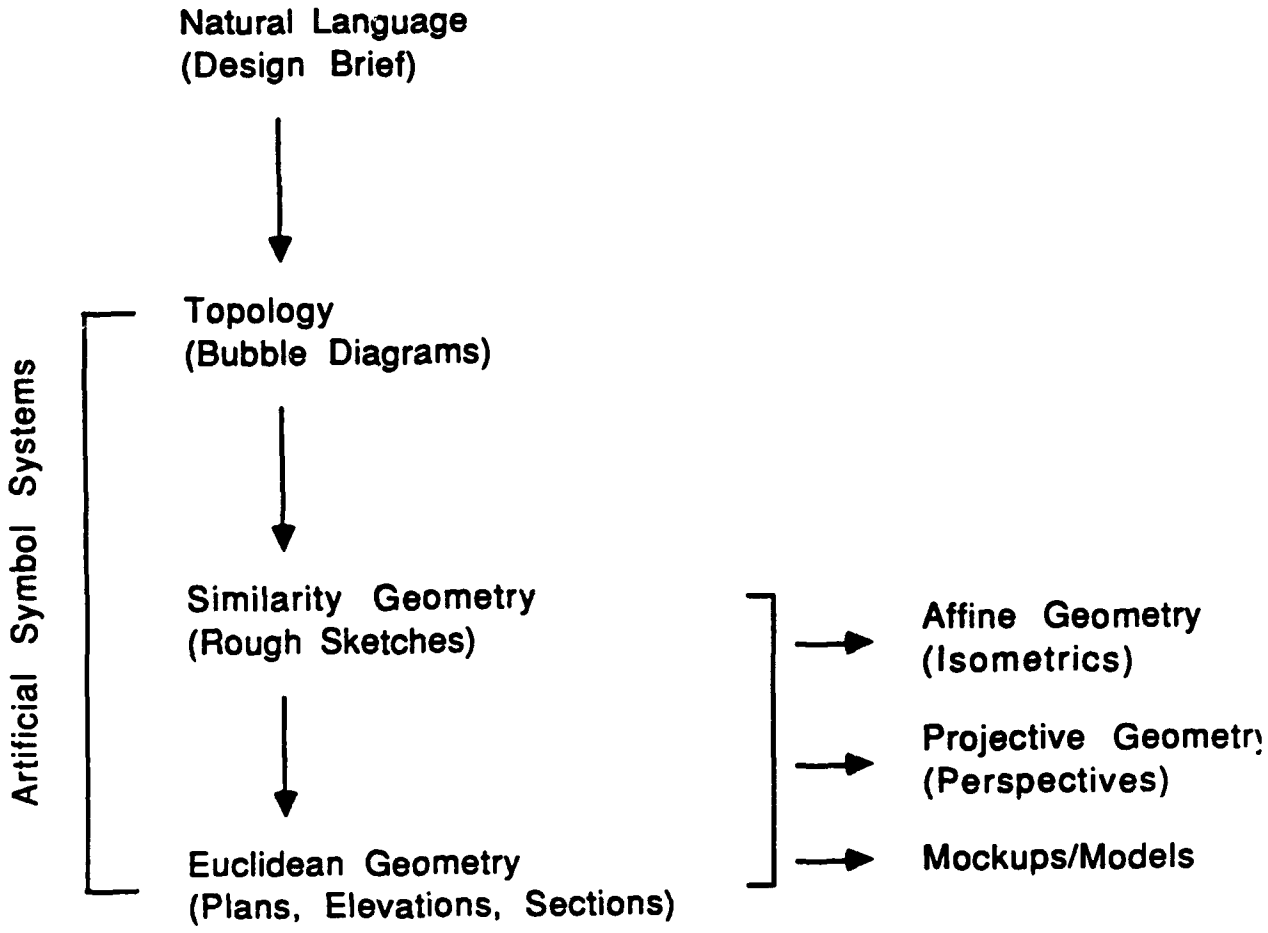


Fig. 4: Symbol Systems Used in Architectural Design

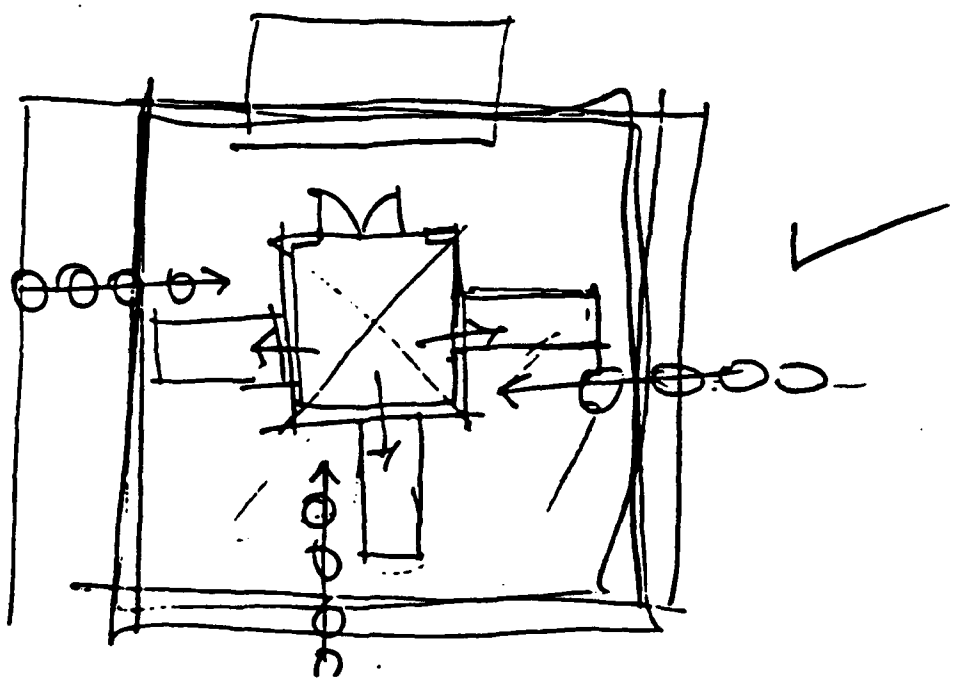


Fig. 5: First Rough Sketch of Floor Plan of Post Office. See text.

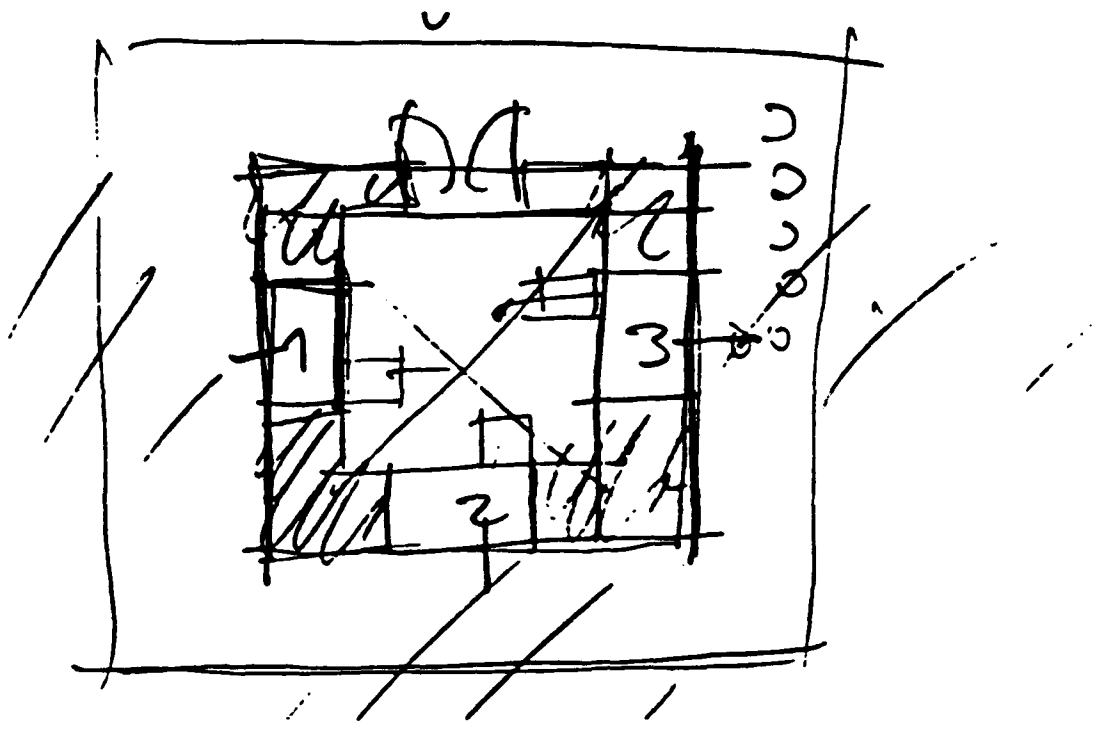


Fig. 6: Second Rough Sketch of Floor Plan of Post Office. See text.

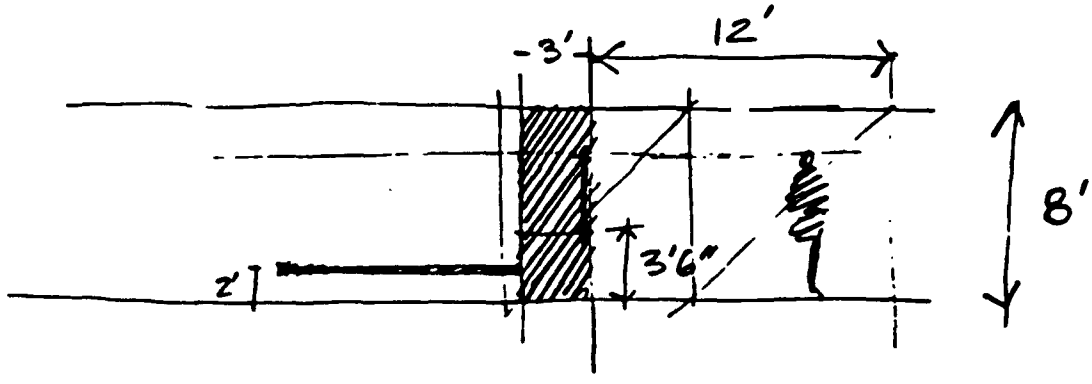


Fig. 7: First Rough Sketch of (Vertical) Section of Post Office. See text.

ATTACHMENT 2

JANUARY 1991 REPORT NO. DPS-3

The Structure of Design Problem Spaces

Vinod Goel and Peter Pirolli

University of California at Berkeley

In press: *Cognitive Science*

Portions of this research were funded by the Cognitive Science Program, Office of Naval Research under contract N00014-88-K-0233, Contract Authority Identification Number NR4422550. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

The Structure of Design Problem Spaces

Vinod Goel
and
Peter Pirolli

University of California, Berkeley

ABSTRACT

It is proposed that there are important generalizations about problem solving in design activity which reach across specific disciplines. A framework for the study of design is presented which (a) characterizes design as a radial category and fleshes out the task environment of the prototypical cases; (b) takes the task environment seriously; (c) shows that this task environment occurs in design tasks, but does not occur in every nondesign task. (d) explicates the impact of this task environment on the design problem space, and (e) demonstrates that, given the structure of the information processing system , the features noted in the problem spaces of design tasks will not all occur in problem spaces where the task environment is vastly different. This analysis leads to the claim that there are a set of invariant features in the problem spaces of design situations which collectively constitute a *design problem space*. Protocol studies are reported in which the problem spaces of three design tasks in architecture, mechanical engineering, and instructional design are explored and compared with several protocols from nondesign problem-solving tasks.

The proper study of mankind is the science of design.

--Herbert A. Simon

1. Introduction

Design is a quintessential cognitive task. The activity of design involves the mental formulation of future states of affairs. The products of design activity are external representations of such possible futures. Creative design is also one of the recognizable features that distinguishes modern humans from other intelligent makers of artifacts (Mellars, 1989; White, 1989; Wynn, 1979, 1981). Design is therefore fundamentally mental, representational, and a signature of human intelligence; features that surely make it an important subject of study in cognitive science.

The study of design in cognitive science is also timely in the context of past and recent developments in the field. Much of the early work on problem solving (e.g, Newell & Simon, 1972) examined performance on well-structured, semantically impoverished tasks -- such as puzzle solving -- having well-defined goals, problem states, and operators. Ill-structured problems, which have ill-defined goals, states, or operators, have been receiving increasing attention. Such research includes studies of software design (Guindon, 1989; Jeffries, Turner, Polson, & Atwood, 1981; Kant, 1985), mechanical engineering (Ullman, Dietterich, & Stauffer, 1988), and the formulation of administrative policy (Voss, Greene, Post, & Penner, 1983). In addition, studies of problem solving in semantically rich domains involving the use of mental models, such as physics (Gentner, & Gentner, 1983), have also increased over the past decade. Design, of the sort we are interested in, is ill-structured in that the tasks involve underspecified goals and operators. The kinds of knowledge that may enter into a design solution are practically limitless. Furthermore, since design inherently consists of the formulation of models of possible states of affairs in the world, it intrinsically involves mental models and a rich set of semantics. Given such trends in the study of problem solving, design is a compelling testbed for further integration and advancement of theories of cognition.

There are also practical reasons for being interested in the study of design. Design activity has major impact on construction and manufacturing costs and capabilities. While it is the case that design accounts for only 3 - 7% of product costs, and that the real money is spent during the manufacturing or construction stage, it is also the case that 80% of manufacturing costs are committed to during the first 20% of the design process (Dixon, & Duffy, 1990). Analysts have noted serious shortcomings in both education and practice of design and have recently stressed that

a lack of a basic scientific understanding of the design process is endangering American industry and productivity (Dertouzos, Lester, & Solow, 1989).

In this paper, we present a framework, theory, and method for the study of design tasks, along with empirical analyses of expert design in three diverse disciplines. Our goal is to address the knowledge and cognitive processes of individual designers, although, clearly, design is often the coordinated effort of many individuals (Curtis, Krasner, & Iscoe, 1988). The framework is intended to encompass theories or models that generalize across design tasks, provide accounts of individual acts of design, and afford computational accounts of moment to moment information processing. In the next section we embed our work in the literature with a very brief outline of past design research and then present our research framework and empirical analyses in subsequent sections.

2. An Overview of Design Research

The development of systematic theories of design has a long history within the design professions, dating back to Leon Battista Alberti's (Alberti, 1450/1988) treatise on architecture. More recently, cross-disciplinary systematication of design was attempted in the *design methodology* movement. In part, this movement was spurred on by the large military and civilian projects of the 1950's and 1960's, such as the development of the Polaris Missile and the moon landing. The design methodology movement responded with a number of *prescriptive* proposals for the systematization of the design process. A number of researchers (Alexander, 1964; Archer, 1969) observed that design had two components: a logical element and a creative element. Both were necessary, but required very different abilities. The basic idea of the design methodology movement was to develop systematic external methods and tools to better carry out the logical analysis, and to unburden the designer to engage in the creative aspects of the problem solving (Cross, 1984).

One of the intellectual outcomes of this research was a consensus among many practitioners that, while the various design professions -- e.g. architecture, engineering, industrial design, urban design, etc. -- differed in particulars, they none the less shared a common core which united them as design professions and differentiated them from nondesign professions such as medicine. This unifying core was thought to reside in the fact that design problem-solving activity involved the following sequence of steps:

1. An exploration and decomposition of the problem (ie., analysis).
2. An identification of the interconnections between the components.
3. The solution of the subproblems in isolation.
4. The combination (taking into account the interconnections) of the partial solutions into the problem solution (i.e. synthesis).

On this basis many researchers concluded that "the logical nature of the act of designing is largely independent of the character of the thing designed" (Archer, 1969, p.76). These prescriptive proposals for revamping design methods gave way to studies involving either the logical analysis of design problems (Alexander, & Poyner, 1966; March, 1976; Reitman, 1964; Rittel, & Webber, 1973; Simon, 1973) or empirical studies (Akin, 1979; Akin, 1986; Darke, 1979; Eastman, 1969) of design as it naturally occurs in the world (or at least the laboratory).

Another line of attack on design problems emanated from information-processing analyses of problem solving. Design problems that require some creativity for their solution were identified as ill-defined or ill-structured problems by Reitman (1964). In other words, such problems initially have underspecified or ambiguous specifications of their start state, goal state, or the function that transforms the start state to the goal state. Simon (1973) argued that there was nothing intrinsic about a problem, per se, that made it ill- or well-structured, but rather that such properties could only be determined by examining the relationship between the problem solver, its available knowledge, and the problem to be solved. Furthermore, he concluded that information-processing accounts were adequate to deal with both ill-structured and well-structured problem solving.

Many recent analyses of design have been carried out in the framework of cognitive psychology or artificial intelligence (Akin, 1979; Akin, 1986; Brown, & Chandrasekaran, 1989; Guindon, 1989; Jeffries, et al., 1981; Kant, 1985; Mostow, 1985; Ullman, et al., 1988; Tong & Franklin, 1989). However, as noted in Goel and Pirolli (1989), much of the cognitive science research on design problem-solving suffers from the following difficulty: Either the research has tended to concentrate on the analysis of discipline-specific design domains and has shied away from characterizations of cross-disciplinary generalizations; or the term "design" has been applied to an increasingly large set of activities that begins to drain the term of substance. Such unlikely tasks as learning (Perkins, 1986) , communication (Thomas, 1978), letter writing, naming, and scheduling (Thomas, & Carroll, 1979) have all been called design activities.

We differ from much of this research in neither believing that (a) design activity characterizations must be discipline specific, nor (b) that design is a ubiquitous activity. There is no more reason to construe every cognitive activity as a design activity than there is reason to construe every cognitive activity as a game playing activity or a natural language generation activity. We assume that there are significant commonalties in the structure of design problems and tasks across the various design disciplines; and there are significant differences in the structure of design problems and nondesign problems. As such, we make a strong commitment to the study of design as a subject matter in its own right, independent of specific tasks or disciplines. We use the term *generic design* to refer to this study. We should note, however, that we are not alone in thinking that there must be interesting generalizations to be drawn across broad classes of problems (e.g., Chandrasekaran 1983; Greeno, 1978).

Another way in which our work differs from other efforts in cognitive science is that we explicitly acknowledge that our studies must emphasize the analysis of design problems and the situations in which they occur. While the importance of analyzing such *task environments* has been stressed in classical theories of human information processing and problem solving (Newell, & Simon, 1972), such analyses are often taken for granted. In particular, researchers dealing with design problems have not let such understanding inform their data analysis and theorizing. We propose to take and develop the notion of design problems and the contexts in which they occur and let it seriously guide our cognitive characterizations.

Our research can therefore be viewed as an integration of two themes. It is an attempt to show that design problem-solving is a natural category of activity and is interestingly different from nondesign problem-solving activity. This attempt draws on the analysis of design problems and situations, and uses the results as both framing and explanatory constructs. In the next section we propose a framework in which to carry out such a study, and subsequent sections describe empirical analysis conducted within the framework.

3. A Framework for Studying Design

In order to study generic design, we need to develop criteria for demarcating and recognizing design problems, and we need some understanding of their common characteristics. Our framework for such an analysis derives from Newell and Simon's (1972) information-processing theory of human problem solving. Basically, a human problem solver is viewed as an information-processing system with a problem. A *task environment* is the external environment, inclusive of the problem, in which the information processing system operates. A *problem space*

is a formalization of the structure of processing molded by the characteristics of the information-processing system, and more importantly, the task environment. A problem space is defined in terms of states of problem solving, operators that move the problem solving from one state to another, and evaluation functions.

Our intuitions about generic design can be formulated in information-processing theory as an hypothesis about the *design problem space*:

> **Design Problem Space Hypothesis**: Problem spaces exhibit major invariants across design problem-solving situations and major variants across design and nondesign problem-solving situations.

Our level of characterization of the design problem space will not be in terms of states, operators and evaluation functions, as is standard in many psychological investigations (Newell and Simon, 1972), but will be stated in a higher-level vocabulary, in terms of a set of invariant features which are common to or characteristic of design problem spaces. Formally, this move to abstraction is comparable to the development of process and data abstractions in computer science (Liskov, & Guttag, 1986).

The basic strategy for explicating the design problem space will be as follows: (a) specify some salient features in the cognitive structure of the designer; (b) specify the salient common features or invariants in the task environment of design problems; (c) show that they constitute a rather unique set of invariants not found in just any arbitrary problem solving situation; (d) let the problem space be shaped by these two sets of constraints; (e) note the structure of the resulting problem space and make "explanatory connections" between this structure and the invariants of the design task environment and the cognitive system; (f) show that the structure of at least some nondesign problem spaces is very different; (g) make the Newell and Simon (1972) argument that, given the structure of the problem solver as a constant across all cognitive activity, any interesting differences across problem spaces of vastly different tasks will be a function of the task environment; and (h) on this basis claim that these features are invariants of design situations and collectively constitute a design problem space. In the following subsections we specify the structure of the information processing system, the structure of the design task environment, and make some predictions about the design problem space.

3.1. The Structure of the Information Processing System

In the classic Newell and Simon (1972) theory of human problem solving, a cognitive system is a simple, relatively unconstrained mechanism. It is basically a physical symbol manipulation system with memory stores (short-term; long-term; external), a processor, sensory receptors, and motor effectors. There are, of course, several more sophisticated accounts in the literature. We view our use of the Newell and Simon (1972) theory, as a way of providing a reasonable first-order approximation of human information processing that is consistent with a wide variety of more recent proposals of human cognitive architectures (Anderson, 1983; Newell, in press). The essential constraints on human information processing proposed by Newell and Simon (1972) that are relevant to our analysis -- the limitations of short term memory, external memory, and the sequential nature of symbolic processing -- would probably show up in many theoretical alternatives. Even a connectionist theory would probably incorporate similar assumptions to address the human limitations in heeding information, and the sequential nature of problem solving activity.

3.2. The Structure of the Design Task Environment

Task environments consist of (a) a goal, (b) a problem, and (c) other relevant external factors (Newell and Simon, 1972). In many studies of problem solving, the emphasis has been on how the structure and content of a *particular* problem gets mapped onto the problem space. In contrast, our explication of the design task environment involves looking beyond the individual problem and specifying the relevant external factors common to all design problems.

Having put the problem thus, there are two difficulties that must be dealt with. First, before one can look at the commonalties across the category of design problems, one must be able to specify what constitutes the category, or at least to identify members of it. Second, having identified the category in some way, one is confronted with the problem of identifying the aspects of the task environment that are relevant. Both of these difficulties have proven to be a notoriously challenging (Goel and Pirolli, 1989). We provide criteria motivated by categorization theory to address the first problem and rely on intuitions, developed through immersion in the discipline of architectural design[1], to generate criteria that address the second problem.

---

[1]First author.

It is our contention that design is too complex an activity to be specified by necessary and sufficient conditions. Rather, design as a category exhibits what Rosch (Rosch, 1978) calls *prototype effects*. Furthermore, it is what Lakoff (Lakoff, 1987) calls a *radial category* --a category in which there is a central, ideal, or prototypical case and then some unpredictable but motivated variations. Given this assumption, one could use a variety of convergent operationalizations to determine the constituent structure of the category of design activity. For instance, if one shows people a list of professions--e.g. doctor, lawyer, architect, teacher, engineer, researcher--and asks which are the best examples of design professions, people will usually pick out the same few cases. In the above list we believe the best examples would be architecture and engineering. We propose to call these central, or prototypical examples of design professions.

Having made this observation we have a nonarbitrary and interesting characterization of design activity. We are now in a position to take a serious look at the task environment of these prototypical design professions and attempt to isolate some interesting common features. We note the following overt features of design task environments (for an earlier approximation of this list, see Goel & Pirolli, 1989):

A) *Distribution of information*. As was initially noted by Reitman (1964) there is a lack of information in each of the three components of design problems. The start state is incompletely specified, the goal state is specified to an even lesser extent, and the transformation function from the start to goal states is completely unspecified.

B) *Nature of constraints*. The constraints on design task environments are generally of two types, (a) nomological and (b) social, political, legal, economic, etc. The latter consist of rules and conventions and are always negotiable. The former consists of natural laws and are never negotiable. However, the constraints of natural law vastly under determine design solutions. Design constraints are rarely, if ever, logical (i.e. they are not constitutive of the task).[2]

C) *Size and complexity of problems*. Design problems are generally large and complex spanning time-scales on the order of days, months, or even years.

---

[2]These important distinctions have not been widely appreciated in the literature. Simon (1973) for example fails to (refuses to) recognize them.

D) *Component parts.* Being large and complex, design problems have many parts. But there is little in the *structure* of design problems to dictate the lines of decomposition. Decomposition is substantially dictated by the practice and experience of the designer.

E) *Interconnectivity of parts.* The components of design problems are not *logically* interconnected. There are however *many* contingent interconnections between them.

F) *Right and wrong answers.* Design problems do not have right or wrong answers, only better and worse ones (Rittel and Webber, 1973).

G) *Input/output.* The input to design problems consists of information about the people which will use the artifact, the goals they want to satisfy, and the behavior believed to lead to goal satisfaction. The output consists the artifact specification. Functional information in many ways mediates between the input and output information. This is a rather standard characterization adapted from Wade (Wade, 1977), and is further discussed in section 4.1.3.

H) *Feedback loop.* There is no genuine feedback from the world during the problem solving session. It must be simulated or generated by the designer during the problem solving session. Feedback from the world comes only after the design is completed and the artifact is constructed and allowed to function in its intended environment. But of course at this point the feedback can not influence the current project, but only the next "similar" project.

I) *Costs of errors.* There are costs associated with each and every action in the world, and the penalty for being wrong can be high (Rittel and Webber, 1973).

J) *Independent functioning of artifact.* The artifact is required to function independently of the designer.

K) *Distinction between specification and delivery.* There is a distinction to be made between the specification of the artifact and the construction and delivery of the artifact.

L) *Temporal separation between specification and delivery*. There is a temporal separation between the specification and delivery or construction of the artifact. The specification precedes delivery.

These are all significant invariants in the task environments of prototypical design situations. Many of them have been noted previously by other researchers. Our claim here is that we can use them as a template to identify other cases of design. To the extent that the task environment of a given problem situation meets or conforms to this template, that problem situation is a prototypical example of a design situation. To the extent that a task environment varies from this template-- by omission of one or more of the requirements--it is a less central case of design activity.

Note that we are not stipulating what is and what is not a design activity. To do that we would have to insist that the 12 task environment characteristics listed above, or some subset of them, constitute necessary and sufficient conditions for design activity. We make no such claim. Rather, all we are suggesting is that we have a template of some salient characteristics common to the task environment of problem situations that are consistently recognized as good examples of design activity. Problem situations in which the task environment fails to conform to this template on one or more accounts are deviations from the central case. In this paper we will only be interested in central cases and thus have no interest in saying how far a problem can deviate from the prototype and still be considered design. Thus we will use the label "design" to refer to situations that closely conform to the prototypical or central cases.

Some problem-solving situations which fit well into the schema are instructional design, interior design, some cases of software design, and music composition. Some tasks that deviate slightly are writing and painting. In these latter cases there is usually no separation between design and delivery. The problem solver actually constructs the artifact rather than specifying it. Some activities that deviate more radically are classroom teaching, spontaneous conversation, and puzzle-solving.

To illustrate how nondesign tasks differ from design, we will examine the task environments of two well-structured problem tasks that have been extensively studied in the literature (Newell & Simon, 1972). The first task is cryptarithmetic, which is a puzzle in which an addition problem is presented, but all digits have been replaced by letters. The task involves solving the addition problem by making hypotheses about the correspondences between letters and digits. The second task is the Moore-Anderson logic task, which is posed as a game in which subjects

transform given symbol strings into goal symbol strings according to certain transformation rules. The game moves are isomorphic to theorem proving by logical inference. The following is a summary comparison of these two tasks to the 12 features of design environment listed above:

A') *Distribution of Information:* In the case of cryptarithmetic, the start state is completely specified. The goal test is also clearly defined. The transformation function is restricted to only two operations, which are specified in advance: (a) assign a digit between 0 and 9 to a letter, and (b) perform addition. In the Moore-Anderson tasks both the start and goal states are completely specified. The set of legal operators is also specified. All the subject has to do is figure out the right order of application.

B') *Nature of Constraints:* In both cryptarithmetic and the Moore-Anderson task the rules are definitional or constitutive of the task. As such they have a logical necessity about them. If one is violated, we are simply not playing the game, but some other game.

C') *Size and Complexity:* In both cryptharithmetic and the Moore-Anderson task, the problems are relatively small and simple. The ones we looked at (Newell & Simon's,1972) took anywhere from 10 to 40 minutes to solve.

D') *Component Parts:* Even though the problems are relatively small, they break down into a number of components. In cryptarithmetic each row (addend) is treated as a component. In the Moore-Anderson task each well-formed formula (wff) constitutes a component part. However, in both cases, unlike the design cases, this breakdown is enforced by the logical structure of the problem.

E') *Interconnectivity of Parts:* In cryptarithmetic the few components (rows) that do exist, are logically interconnected. That is, there exists the possibility that any row will sum to greater than 9 and affect the next row. In the Moore-Anderson task there are logical connections between the wffs but they are not obvious. In fact, the task is substantially to discover these interconnections.

F') *Right and Wrong Answers:* In the cryptarithmetic problems we examined there was only one definite right answer. All other answers were simply wrong. In the Moore-Anderson task the answer consists of the right sequence of operators. While there may be a set of right sequences (rather than a single one), all others are definitely wrong. Also, it is clear when one has a right sequence.

G') *Input/Output:* In cryptarithmetic the input is limited to letters, numbers, and the rules of the game. The output is a particular sequence of numbers. In the Moore-Anderson task the input consists of the wffs, and the operators. The output consists of the sequence of lawful operator applications sufficient to transform the premise wffs into the conclusion wff.

H') *Feedback Loop:* In cryptarithmetic there is genuine feedback after every operation (i.e. substitution and summation). It is, however, local feedback and the final decision needs to satisfy global constraints. In the Moore-Anderson task there is local feedback after every operator application. But it is of limited value. More useful feedback comes after sequences of operator applications.

I') *Costs of Errors:* In both cases, the cost of error is negligible in the sense that a wrong answer may cause the subject some embarrassment, but it will not affect the lives of third party "users."

J') *Independent Functioning of Artifact:* Not applicable.

K') *Distinction between Specification and Delivery:* Not applicable.

L') *Temporal Separation of Specification and Delivery:* Not applicable.

The reader will note that these nondesign task environments differ from the design task environment on all 12 features listed above. Incidentally, they also differ from each other in respect to features A' and F'. But the consequences of this latter difference will not be pursued here. The consequences of the differences between design task environments and nondesign task environments will be considered in some detail in a later section.

This is not to suggest that all nondesign task environments will be so radically different. For instance, an intermediate case is provided by the task environment of some mathematical problems, such as those presented in Schoenfeld (Schoenfeld, 1985). However, we have purposefully chosen to contrast the design cases with cases which deviate radically. The sharp contrast makes the distinctions clearer. If something substantive is uncovered, we can move onto the subtler comparisons which will be involved in the less deviant cases. If nothing interesting emerges in these clear cut cases, nothing will emerge in the subtler cases.

3.3. The Structure of Design Problem Spaces

The following is a list of a dozen invariants found in the structure of the design problem spaces we examined. Each can be explained or justified by an appeal to the structures of the information-processing system and design task environment as articulated above.

1. *Problem structuring*. The lack of information in start states, goals states, and transformation functions will require extensive problem structuring before problem solving can commence.

2. *Distinct problem-solving phases*. Design problem solving can be further subcategorized into three interestingly distinct phases, preliminary design, refinement, and detail design.[3] This is probably due to the size and complexity of problems and the many different types of information and levels of detail that need to be considered.

3. *Reversing direction of transformation function*. Since the structure of the task is not well specified in advance, and the constraints are nonlogical, the designer has the option to reverse the direction of the transformation function by transforming the problem to one for which he may already have a solution, or into a problem for which the solution is in some way more effective or desirable.

4. *Modularity/decomposability*. Given the size and complexity of design problems and the limited capacity of short term memory one would expect decomposition of the problem into a large number of modules. However, given the fact that there are few or no logical connections between modules but only contingent ones, one would expect the designer to attend to some of these and ignore the others.

5. *Incremental development of artifact*. Interim design ideas are nurtured and incrementally developed until they are appropriate for the task. They are rarely discarded and replaced with new ideas. The principle reasons for this would be the size and complexity of problems and the sequential nature of the information processing system, and the fact that there are no right or wrong answers.

---

[3]There is nothing deep about there being three phases rather than n phases. Designers use these three phases to talk about their process. We too found them useful for our purposes. It would certainly have been possible to do a finer-grained or coarser-grained individuation.

6. *Control structure.* Designers use a *limited commitment mode control strategy* which enables the generation and evaluation of design components in multiple contexts.

7. *Making and propagating commitments.* Since design plans and specifications have to be produced in a finite amount of time, and have to be interpretable by a third party, designers have to make, record, and propagate commitments.

8. *Personalized stopping rules and evaluation functions.* Because there are no right or wrong answers and direct feedback is lacking, the evaluation functions and stopping rules the designers use will be personalized (i.e. derived from personal experience and immersion in the profession).

9. *Predominance of memory retrieval and nondemonstrative inference.* Since there are very few logical constraints on design problems, deductive inference plays only a minimal role in the problem solving process. Most decisions are a result of memory retrieval and nondeductive inference.

10. *Constructing and manipulating models.* Since design typically occurs in situations where it is not possible to, or too expensive to directly manipulate the world, designers usually manipulate representations of the world. (We only get one run on the world, whereas we can get as many runs as we like on models of the world.)

11. *Abstraction hierarchies.* The qualitative difference in the input and output information and the several distinct problem solving phases result in orthogonal abstraction hierarchies.

12. *Use of artificial symbol systems.* Given the size and complexity of problems, the need to construct and manipulate external models, and the use of abstraction hierarchies, designers will make extensive use of artificial symbol systems.

The first six of these invariants will be discussed and illustrated with data in section 5.0. A lengther and more complete discussion of all twelve invariants is available in Goel (1991). But before begining the discussion we present our methodology and data analyses in the next section.

4. Database and Coding Scheme

Our empirical studies have focused largely on the analysis of verbal protocols from expert designers in various disciplines. This has required the development of a rather complex protocol analysis scheme that can be tied to our framework for the design problem space. In addition to developing analyses to find commonalties across design tasks, we have also been interested in exploring how design tasks might differ from nondesign tasks. Consequently, we have also performed analyses of nondesign verbal protocols that were readily available in the literature. Altogether, we have accumulated a database of 12 design protocols and six nondesign protocols. Both sets are described below and we present detailed analyses of three of the design protocols and two of the nondesign protocols.

4.1. Design Protocols

Twelve design protocols, of approximately two hours each were collected from experts in the disciplines of architecture, mechanical engineering, and instructional design. While we have examined and coded all 12, we will restrict our discussion here to one from each discipline. The three were selected one the basis of (a) the completeness of the artifact specification produced by the individuals, and (b) the fluency of the subjects' verbalization.

4.1.1. Subjects

One of the protocols was produced by a subject (Subject S-A) performing our architecture task . Subject S-A was a Ph.D. student in the Department of Architecture at the University of California, Berkeley, who volunteered to participate in the study. Subject S-A had six years of professional experience, but had never designed a post office. A second protocol was produced by Subject S-M in solving our mechanical engineering task. Subject S-M was a Ph.D. student in the Department of Mechanical Engineering at Stanford University, and also volunteered to participate in the study. Subject S-M had three years of professional experience including working for a firm in Italy designing bank teller machines. The third protocol was collected from Subject S-I on an instructional design task. Subject S-I was a professional instructional designer working for a large multinational corporation who also volunteered to participate in the study. Subject S-I had over 10 years experience in designing technical training material, mainly on the operation and servicing of office machinery and systems. Subject S-I was also familiar with the text-editor that was the focus of his design task.

4.1.2. Task Descriptions

- *Architecture.* The architecture task involved the design of an automated post office (where postal tellers were replaced by automated postal teller machines) for a site on the UC Berkeley campus. Subject S-A was given two documents: an outline of the experiment procedures and a design brief which motivated the need for the post office and specified the client's needs and requirements. The site was visible from the place of the experiment.

- *Mechanical Engineering.* The mechanical engineering task involved the design of an automated postal teller machine for the above post office. Subject S-M was given three sets of documents: the experimental procedures, the design brief, and the post office design generated by the architect.

- *Instructional Design.* The instructional design task involved the design of a self-contained instructional package to teach secretaries how to use the Viewpoint text-editing software running on Xerox Stars. Subject S-I was given three documents, the experimental procedures, the design brief, and the Viewpoint reference manual.

Each designer was asked to talk aloud as he or she proceeded with the task and was encouraged to probe the experimenter for further information and clarification as necessary. The experimenter answered questions posed by the subject but at no time initiated questions or engaged the subject in conversation. The sessions were taped on one or more video recorders and all written documents were collected for later analysis.

The tasks were complex real-world problems with most of the features specified earlier for design task environments. They however differed in two respects from the characteristic features of full-scale design tasks, and both of these differences were necessitated by the logistics of data collection. In our particular experimental situation there was no substantive penalty for being wrong or proposing an inferior solution, as there would be in the world at large. Also, while the tasks we gave our subjects each required on the order of weeks to months for complete specification of the artifacts, we asked them to restrict their sessions to approximately two hours. As a result we received solutions specified to an incomplete level of detail.

4.1.3. Protocol Coding Procedures

The verbal protocols were first transcribed and cross-referenced with the written and drawn documents (henceforth referred to as marks-on-paper). The transcribed protocols were divided into utterances or statements, and each statement was coded. Collections of statements were aggregated into units called *submodules* and *modules*. The utterances in modules or submodels focussed on particular components or subcomponents of the artifact. A aggregation of modules and submodules identified sections of protocol that corresponded to different *phases of design development*. In this subsection we specify our criteria for individuating statements and the coding scheme.

4.1.3.1. Individuating Statements

Following previous outlines of verbal protocol analysis (Ericsson, & Simon, 1984), our goal was to divide the protocols into statements that conveyed a single thought, expression, or idea. There are at least two ways of doing this. One is to individuate by content cues such that statements are demarcated by shifts in topic or new points being made about the topic. A second way of demarcating statements is to use noncontent cues such as pauses, phrase and sentence boundaries, and and the making and breaking of contact between pen and paper. We looked at both of these ways of demarcating statements and applied whichever one provided the finer-grained individuation. The mean duration of statements was approximately eight seconds. The mean number of words per statement was about 15 words.

4.1.3.2. Coding Scheme

Our protocol coding scheme was similar in spirit, but not detail, to schemes employed in other recent studies of design (Greeno, Korpi, Jackson, & Michalchik, 1990; Ullman, et al., 1988). Each statement was coded along several dimensions (see the Goel (1991, Appendix D) for a more complete description of the scheme along with illustrative examples). Figure 1 illustrates how each statement could be assigned to one of four categories used to identify the *general focus* of a designer's activity. *Experimental task* statements were statements concerned with the experimental design and setup. *Monitor* statements indicated metacognition, or reflection about design methods. These were statements in which a subject reviewed or commented upon the problem solving process itself. Three different types of monitoring statements were identified: explicit mentions of *design operators*, statements about *design methodology*, and statements about *book-keeping*. *Design development* statements were statements that advanced the specification of the artifact.

Design development statements were further divided into: (a) *problem structuring* statements, which generated or solicited information that further structured the problem, or (b) *problem solving* statements, in which the design specification was advanced in some way. Problem solving statements were divided into: (a) *preliminary-design* statements, which were the initial specifications of a design solution, (c) *refinement statements*, which elaborated an established design element, and (d) *detail* statements, which served to finalize the form of some design element. These labels are rather standard terms that designers routinely use to talk about various phases of design development. They are of course relative to the available time and the degree of completeness of the design specification. But none the less, given the final output of the designer, it is possible to trace the development and segment it into phases.

We also coded all design development statements according to the *aspects of design development* referred to in the statements. This subcategorization was orthogonal to the breakdown of design development statements presented in Figure 1, and differentiated the type of content attended to by the designer. For each statement we identified whether the content of the statement referred to: *people*, *purposes*, *behavior*, *function*, *structure*, and *resources*. As with the problem solving categories, these subcategories are also quite standard in the design literature. The ones we employ are adopted from Wade (1977). Briefly, the intuition behind these terms is that *artifacts* are designed to perform certain *functions* which are calculated to support certain *behaviors*, which help in the realization of certain *purposes* held by *people*. This categorization provides a chain linking users to artifacts and recognizes that each intermittent step needs to be considered. To these categories we have added *resource* (e.g. time, money, manpower, etc.). It should be noted that these are not disjunctive categories. A single statement can fall into more than one category.

In addition to coding the general aspects of the design being attended to, we also aggregated statements into episodes of closely related statements united by a focus on some particular component of the designed artifact. We called these aggregates *modules* and *submodules*. Unlike the codings discussed so far, the module coding scheme was dependent on the particular design tasks and the particular individual solving the task.

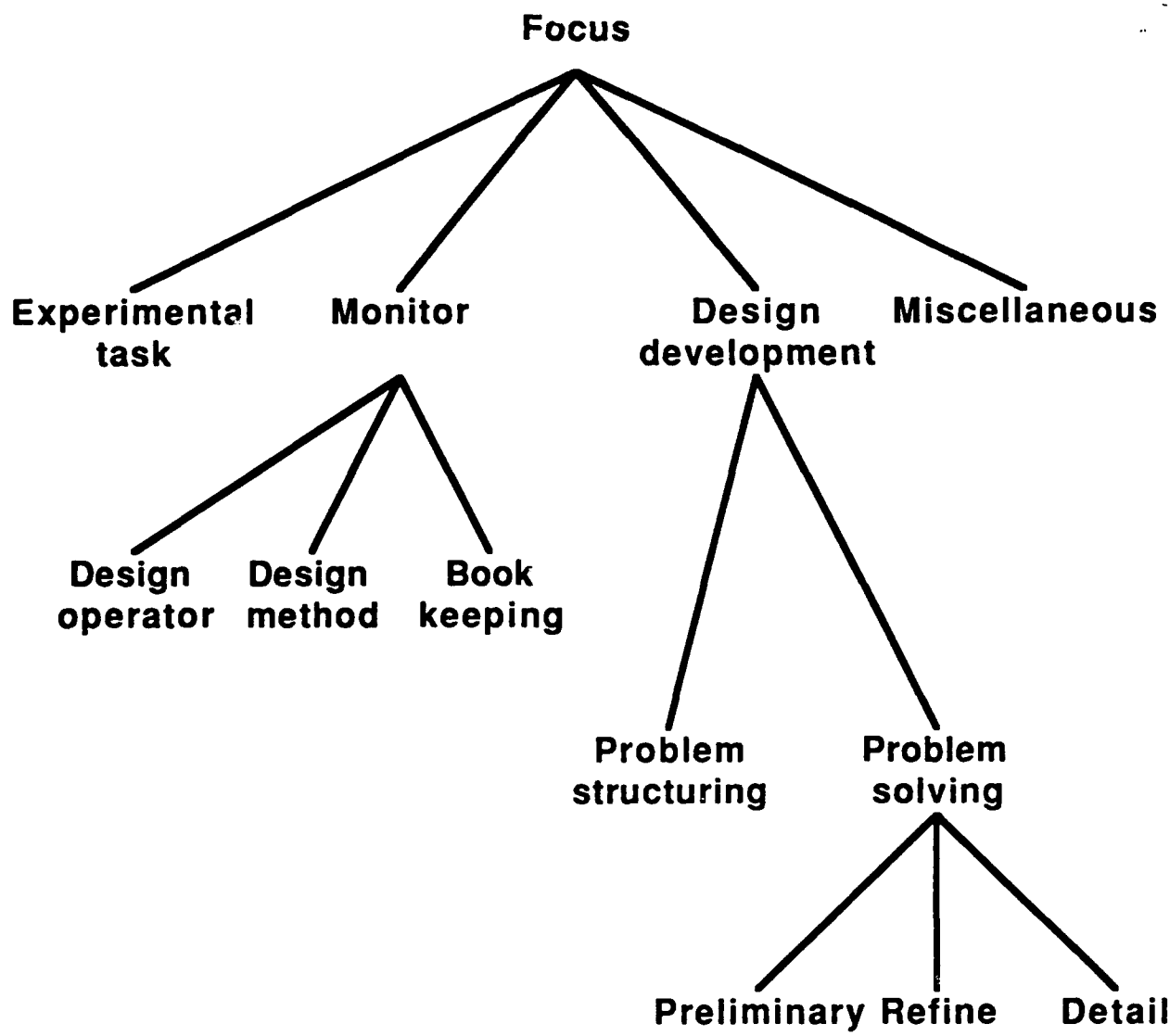


Figure 1. Categories of general focus in protocol coding scheme.

Finally, for each statement, we coded a set of information about the particular problem-solving step being executed by a designer: (a) the *operator* applied, (b) the *content* to which it was applied, (c) the *mode* of the output, and (d) the *source* of knowledge used (see Figure 2). Operators were a labeling of statements by the function they served in the problem space. Although we made no theoretical commitment to any specific set, we found the eleven noted in Figure 2 adequate for our needs. The mode of output of a statement was encoded as either *verbal* or *written*. Each statement was also coded for the source of knowledge for the statement. The four categories used were the *experimenter*, the *design brief*, *self* (retrieved from long-term memory), and *inferred* (deductively) from the information existent in the problem space.

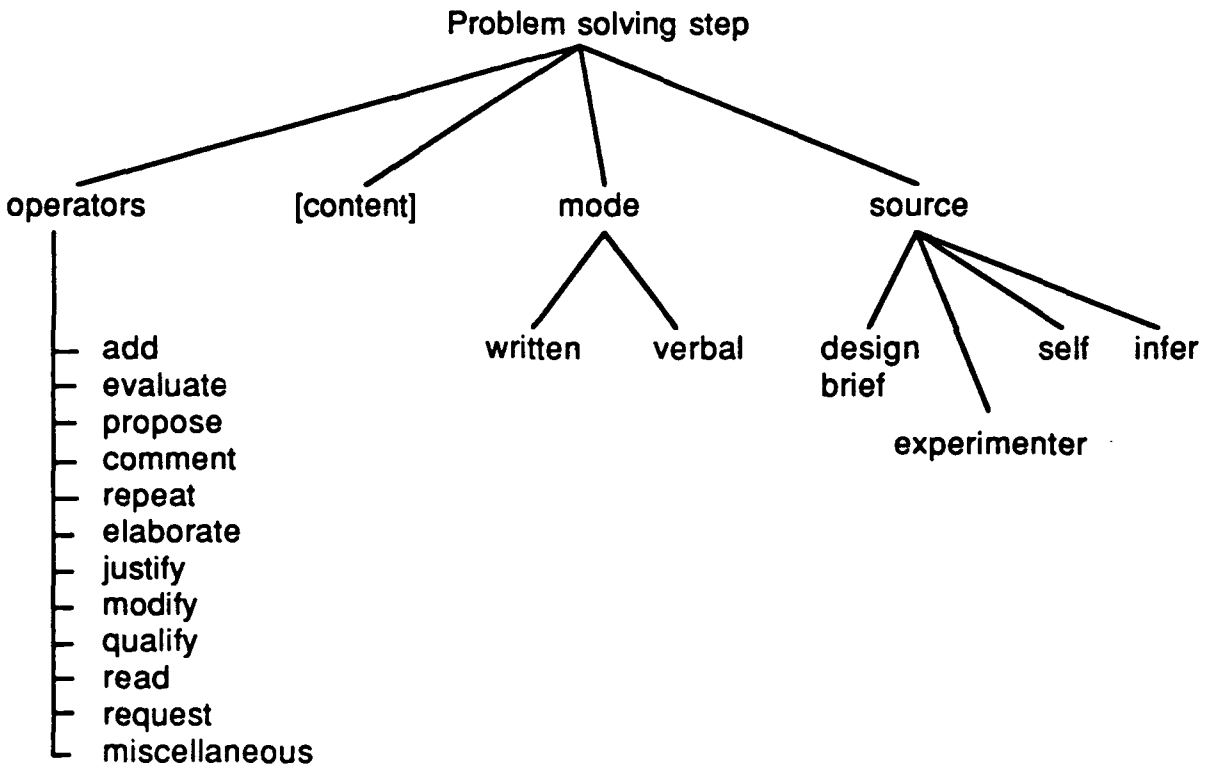


Figure 2. Localized problem-solving step categories in protocol coding scheme.

Although the complete coding scheme was complex, each part of the scheme provided a different view onto the unfolding of the design process and the evolution of the design itself. In the next subsection, we describe how the scheme was modified to code protocols collected in nondesign tasks.

4.2. Nondesign Protocols

Our six nondesign protocols were gathered from several published sources. They include mathematics, cryptarithmetic, and the Moore-Anderson tasks, and range in duration from 15 minutes to 40 minutes. The two mathematics protocols were taken from Schoenfeld (1985, Appendices 9.1 and 9.5). These two were selected from several in Schoenfeld (1985) because they were the only ones which approximated our experimental setup of single subjects giving uninterrupted and unprompted protocols. The two cryptarithmetic and two Moore-Anderson task protocols were chosen from Newell and Simon (1972, Appendices 6.1, 7.1, 9.2 and 10.1). These protocols were chosen on the basis of their duration.

Each of these six nondesign protocols was analyzed and coded. However, keeping with our strategy of contrasting extreme cases, we discuss and compare only the cryptarithmetic and Moore-Anderson task problem spaces with the design problem space.

4.2.1. Subjects

The subjects in both the cryptarithmetic and Moore-Anderson tasks were undergraduate college students.

4.2.2. Task Descriptions

The cryptarithmetic tasks for both subjects (NS6.1 and NS7.1) involved solving the following puzzle:

```
   DONALD        D = 5
 + GERALD
 -----------
   ROBERT
```

in which each letter stands for a specific digit and the digits associated with ROBERT are the sum of the digits associated with DONALD + GERALD, and it is given that D = 5. Cryptarithmatic problems are basically constraint satisfaction problems.

The Moore-Anderson task is a string transformation task, isomorphic to deductive inferences in propositional logic using logical inference schemas. The task for Subject NS9.2 in Newell and Simon (1972) was

L1: (R --> ~P) . (~R --> Q)
L0: ~(~Q . P)

where L1 was a premise and L0 was a goal. The transformation rules are specified in Newell and Simon (1972, p. 406). Using the same transformation rules, the task for Subject NS10.2 in Newell and Simon (1972) was:

L1: P . (Q . R)
L2: ~(P --> T) --> ~(P . Q)
L0: T . T

where L1 and L2 were premises and L0 was the goal.

4.2.3. Protocol Coding Procedure

The protocols were reproduced from their original source and recoded with a subset of the scheme devised for the design protocols that was modified as follows:

1. As with the design protocols, design development statements were differentiated into problem structuring and problem solving statements. However, the problem solving statements were not further differentiated into the various design phases.

2. The aspects of design development category was eliminated.

3. The mode of output was not coded.

The first two changes were necessitated by the data. There is no interesting sense in which the nondesign data was amenable to a further break down of the problem-solving category into subcategories and the aspect of design development category was inappropriate. Both of these issues will be discussed below. There was not enough information available in the published sources to code for the mode of output.

5. The Design Problem Space

In this section, we list and briefly discuss the some of the more interesting invariants in the design problem space. In each case we ask the following three questions: (a) what is the phenomenon, (b) why does it occur, and (c) what is the supporting data. We will also consider if and in what form the phenomenon transpires in non-design problem spaces.

5.1. Problem Structuring

Problem structuring is the process of drawing upon our knowledge to compensate for missing information and using this knowledge to construct the problem space (Simon, 1973). It occurs for the obvious reason. Design problems are incompletely specified but the specification of a problem space requires complete information about start states, goal states, operators, and evaluation functions. In Goel and Pirolli (1989) we examined the form and organization of some

of this knowledge and how it was applied. Here we want to note the extent and location of problem structuring phases and also say a few words about how they differ from problem solving.

The first thing to note is that problem structuring accounted for approximately 25% of the statements devoted to design development in our design protocols. In our three protocols it ranged from a low of 18% to a high of 30% (for a mean of 24%). In contrast, only 0.3% of the nondesign protocols were devoted to problem structuring. The second point is that problem-structuring statements occurred mainly at the beginning of the task, where one would expect it, but also reoccurred periodically as needed. Figure 3 shows the temporal distribution, aggregated over 5 minute intervals, of the problem-structuring and problem solving phases for subjects S-A, S-M, and S-I.

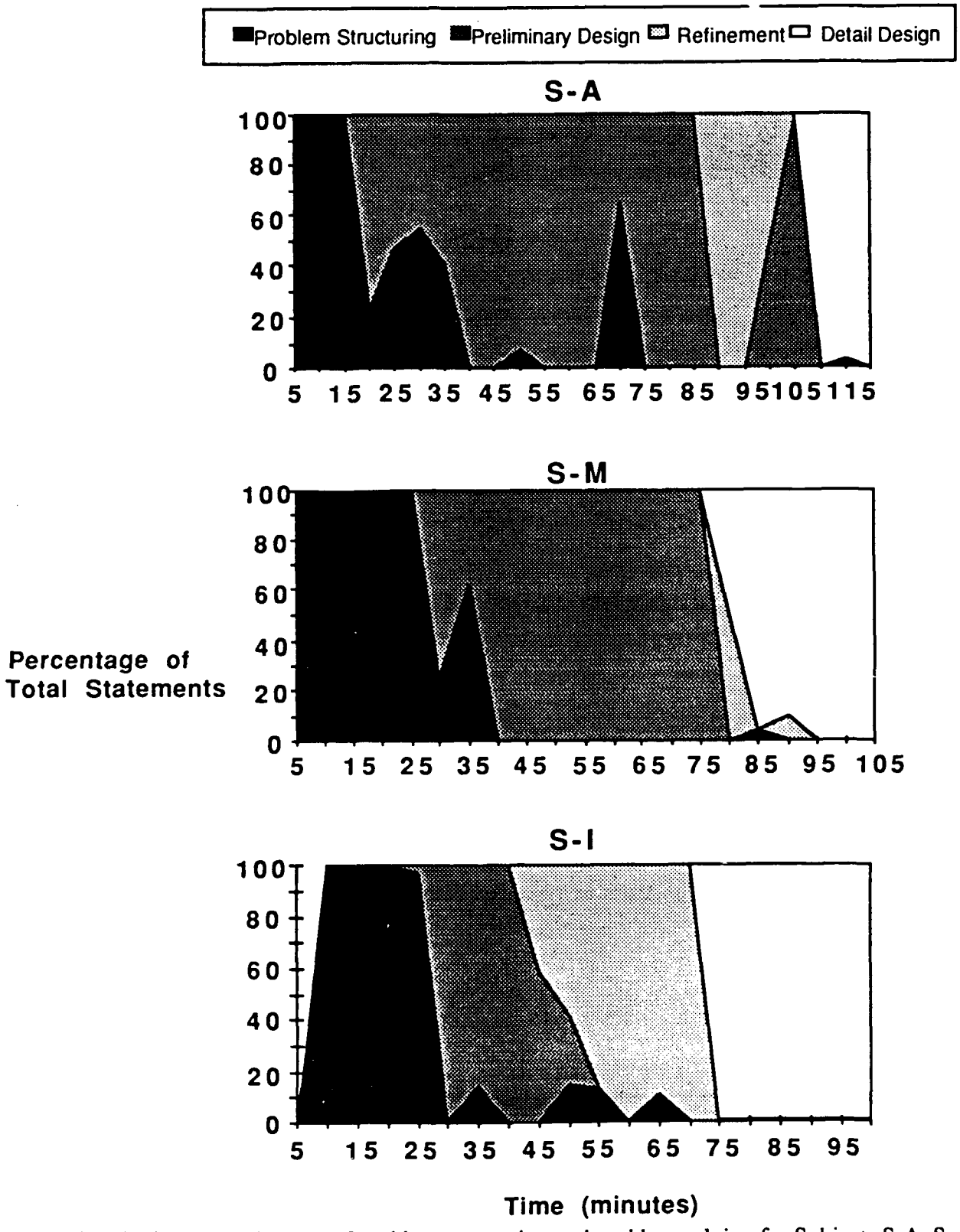


Figure 3. Distribution and extent of problem structuring and problem solving for Subjects S-A, S-M and S-I (aggregated over five-minute intervals).

Although problem structuring is a widely recognized concept, it is often unclear as to how it differs from problem-solving. In our design protocols, we noted several phenomena that appear to differentiate problem structuring activities from problem solving activities:

1. *Aspects of the design considered:* Table 1 presents a breakdown of the aspects of design attended to by subjects across the design development phases. In general, subjects produced proportionately more statements about people, the purposes of the artifact, and resources during the problem structuring phases than in problem solving phases (preliminary design, refinement, and detail design). In contrast, statements about the structure or function of the artifact were more prominent in the problem solving phases than in the problem structuring phases. These results suggest that problem structuring in design is associated with attention to how the artifact may be used and what is available to form it, whereas problem solving is associated with attention to specification of the function and form of the artifact.

2. *The primary source of knowledge:* The client and design brief were important sources of knowledge during problem structuring but not during problem solving (Table 2). Problem structuring is associated with bringing new information into the problem space.

3. *The degree of commitment made to output statements (as evidenced by the quantity and character of written output):* There was a higher percentage of verbal-only statements generated during problem structuring than during problem solving (Table 3). To some extent, this may indicate that written output indicates some degree of commitment to particular design decisions, whereas purely verbal statements about the design indicate less commitment.

4. *Operators*: There was a higher percentage of add and propose operators in the problem structuring phase than the problem solving phase. 42.3% of the operators applied during the problem structuring phase were add operators. This was systematically reduced to 36.7%, 35.3%, and 32% during the preliminary design, refinement, and detailing phases respectively. Similarly, propose operators which accounted for 10.7% of the operators applied during problem structuring, were systematically decreased to 10%, 7.6%, and 6.7% during the preliminary design, refinement, and detailing phases respectively.

Table 1
Proportion of statements made in each design development phase about various aspects of a design.

		Problem Solving		
	Problem Structuring	Preliminary Design	Refinement	Detail Design
Subject S-A				
People	.14	.13	.06	.01
Purpose	.14	.02	.00	.00
Resource	.11	.02	.00	.03
Behavior	.08	.11	.01	.01
Function	.10	.18	.10	.05
Structure	.43	.54	.83	.90
Subject S-M				
People	.22	.07	.22	.06
Purpose	.04	.01	.00	.00
Resource	.09	.03	.00	.00
Behavior	.10	.11	.03	.09
Function	.26	.35	.06	.10
Structure	.29	.43	.69	.76
Subject S-I				
People	.33	.10	.00	.00
Purpose	.16	.02	.00	.00
Resource	.26	.00	.00	.00
Behavior	.04	.02	.00	.00
Function	.00	.15	.48	.33
Structure	.20	.71	.52	.67
Combined Subjects				
People	.22	.11	.04	.02
Purpose	.10	.02	.00	.00
Resource	.14	.02	.00	.01
Behavior	.08	.09	.01	.03
Function	.14	.23	.32	.19
Structure	.32	.53	.63	.75

Table 2
Breakdown of knowledge sources for statements made in each design development phase.

		Problem Solving		
	Problem Structuring	Preliminary Design	Refinement	Detail Design
Subject S-A				
Design Brief	.09	.00	.04	.01
Experimenter	.13	.01	.00	.00
Self	.77	.96	.88	.93
Inferred	.01	.03	.08	.06
Subject S-M				
Design Brief	.11	.00	.00	.00
Experimenter	.44	.04	.00	.00
Self	.44	.93	1.00	1.00
Inferred	.00	.03	.00	.00
Subject S-I				
Design Brief	.38	.00	.00	.00
Experimenter	.14	.05	.01	.00
Self	.47	.95	.99	1.00
Inferred	.01	.00	.00	.00
Combined Subjects				
Design Brief	.21	.00	.01	.00
Experimenter	.25	.03	.00	.00
Self	.53	.95	.96	.98
Inferred	.01	.02	.03	.02

Table 3
Output mode associated with statements in each design development phase.

		Problem Solving		
	Problem Structuring	Preliminary Design	Refinement	Detail Design
Subject S-A				
Verbal	.86	.84	.58	.67
Written	.14	.16	.42	.33
Subject S-M				
Verbal	.89	.55	.52	.66
Written	.11	.45	.48	.34
Subject S-I				
Verbal	.80	.55	.60	.19
Written	.20	.45	.40	.81
Combined Subjects				
Verbal	.88	.77	.61	.71
Written	.12	.23	.39	.29

5.2. Distinct Problem Solving Phases

In addition to the distinction between problem-structuring and problem-solving noted above, there is a further differentiation of problem-solving into several distinct phases. We have subcategorized problem-solving into preliminary design, refinement, and detail design. As noted earlier, these categories are quite standard among designers. As might be expected, these phases were generally engaged in sequentially by our subjects, starting from preliminary design, passing through refinement, and ending with detail design; though it was not unusual for a subject to return to an earlier phase as previously unnoticed aspects emerged (Figure 3). There was however considerable variability between subjects as to the amount of time devoted to each phase.

The three problem solving phases differ at least in terms of the following three respects:

1. *Aspects of the design considered*: There was a steady decrease in the consideration of people, purpose, and resource aspects of design development from preliminary to detail design and a corresponding increase in the structural aspect (see Table 1). The behavior and function aspects on average seemed to stay relatively constant across the three phases.

2. *The primary source of knowledge*: There was still some input from the client and/or design brief at the preliminary design stage, but it disappeared by the detailing stage (see Table 2).

3. *The degree of commitment made to output statements (as evidenced by the quantity and character of written output)*: There was a steady increase in the number of verbalizations that were committed to paper as subjects progressed from preliminary design, through refinement, to detail design (see Table 3). There was also an increase in the degree of explicitness and detailing in the written or drawn material (see Goel (1991) for illustrations). We take both of these as evidence of increasing commitment to the emerging design.

It is our conjecture that the distinct phases result from not only the size and complexity of the problems, but perhaps more importantly, from a combination of the different types of

information that must be considered during the session (i.e., people, purposes, behavior, function, and structure) and the different levels of detail at which it is considered.

Again, the situation was quite different for the nondesign protocols. Instead of problem solving being comprised of distinct phases of different activities, it was comprised of cycles of the same basic activity. In nondesign problems, the subject is searching for a solution. If it is not on the path being searched, one must back up and start down another path. The activity one engages in as one goes down each path is basically the same. This is demonstrated by data that will be discussed in a later subsection.

5.3. Reversing the Direction of Transformation Function

The designer naturally interprets the problem situation through personal experiences and biases. But in addition to this, designers will occasionally stop and explicitly try to change the problem situation so it more closely fits their expertise, knowledge, and experience. This involves manipulating both the problem constraints and the client's expectations. We call this *reversing the direction of the transformation function* because rather then transforming initial problem states to a goal state, the designer can negotiate changes to the goal state that experience suggests are more easily achievable, or perhaps might lead to a more effective design solution.

An example of an (unsuccessful) attempt to explicitly change the goal state specified by a design brief was provided by the following negotiation sequence. Subject S-A was standing on a ninth floor balcony and had a birds-eye view of the small triangular site he had been given for the proposed post office. He was not content to just build a post office but wanted to redesign the whole area.

S-A: So, given the fact we have that triangle over there as a limit. And I cannot exceed that I suppose?

E: Right, that, that...

S-A: I have to take that for granted?

E: I, I would think so.

S-A: That's the boundary of. You do not allow me to, to exceed in, in my area of intervention?

E: No, I think you should restrict it to that.

S-A: So, I am constrained to it and there is no way I can take a more radical attitude. Say, well, look, you are giving me this, but I actually, I, I'd come back to the client and say well look, I really think that you should restructure actually the whole space, in between

> the building. I'd definitely do that, if that was the case. You come to me as a client, and come to me with a triangle alone, I will give you an answer back proposing the whole space. Because, I, I think the whole space should be constructed. So, that there is an opportunity to finally to plan and that space through those, ah, this building, open up Anthropology and, and plan the three buildings together. So, as to really make ah, this ah, a more communal facility....

The reasons why such episodes occur are clear: (a) the problem are incompletely specified, and (b) design constraints are nonlogical and therefore manipulable.

Notice such a sequence simply could not (and does not) occur in nondesign problem spaces where the problem constraints completely specify the problem, and indeed are constitutive of the problem. If it did occur--if the subject requested a change in the problem parameters--we would simply say that he could not do the assigned problem and was changing it to a different problem.

5.4. Solution Decomposition into Leaky Modules

A number of researchers have noted the important role played by decomposition in dealing with complexity (Alexander, 1964; Simon, 1962, 1973b, 1977). However, there is considerable disagreement as to the extent and structure of decomposition of design problems. Some researchers assume strict tree-like decompositions (Alexander, 1964; Brown and Chandrasekaran 1989). But Alexander (1965) argued that "A city is not a tree; it is a semi-lattice." There has also been some discussion as to the extent of interconnections or the encapsulation of the modules.

In Goel and Pirolli (1989) it was noted that designers decomposed the design solution into "leaky modules" (i.e. sparsely connected modules) and had two main strategies for dealing with these interconnections: (a) they either blocked the leaks by making functional level assumptions about the interconnected modules; or (b) they deferred further development of current module while they attended to an interconnected module. It was also stressed that partial interconnectivity (as opposed to total connection or total disconnection) was a genuine phenomenon and had to be taken seriously.

The decomposition of a design into maximally independent units ameliorates the difficulties faced by a limited capacity human information processor dealing with typically large and complex design problems. Yet it is also a fact of the world that artifactual objects and processes are

composed of entities that are in fact related to one another in complex ways. Somehow, the design process has to decompose a design to reduce attentional loads, yet remain attendant to possibly important interconnections between decomposed modules.

In this study we investigate the density and distributions of these interconnections. We divided the protocols into modules (see section 4.0) and used the mentioning of one module inside another module as an indication of interconnections between the modules. Here we report the findings for subject S-A.

Subject S-A decomposed his solution into 34 modules clustered in four larger groups. (We will refer to these 34 modules as submodules and use the term 'module' to refer to the four larger groups.) The four groups (or modules) were "Site," "Building," "Services," and "Automated Postal Teller Machine." The submodules within the Site module were things like trees, illumination, circulation, etc. The submodules within the Building modules included, mail storage, configuration of plan, roof, location of equipment, etc. The Services modules included items such as number of machines, mail pickup, number of people and service times, etc. The Automated Postal Teller Machine module included things such as machine components, interface, and stamping procedure.

Given these 34 submodules there are 1,122 logically possible interconnections that can be made. We found that on average, 7.4% of these connections were actually made. Furthermore, there is, as might be expected, a difference in the density of connections between the four major groups and between the submodules internal to the groups. The former connections are considerably denser than the latter (13.4% vs. 4.5%). These results support claims of the *near* decomposibility of design solutions.

To compare these results with the the nondesign problem spaces we did the same analysis on several nondesign protocols. In the case of cryptarithmitic we found that subjects decomposed the task into modules corresponding to columns of letters. Since the problems only had six columns there were only six modules (as compared to over 30 for each of the design tasks). But while the actual number of modules were substantially fewer, the density of interconnections between modules was considerably greater. Using the same procedure as above we find that the density of interconnections for subject NS6.1 is greater than that of the design problem space. In the cryptarithmetic problem space of subject NS6.1, 20% of the possible connections were made, as opposed to an average of 7.4% for the design problem space of subject S-A.

The denser interconnectivity of the cryptarithmetic modules is exactly what one would expect given the fact that it is designed as a multiple constraint satisfaction problem and all the constraints are logical (i.e. they must be attended to). This is perhaps why such problems can have relatively few modules and still be very challenging. The reason design problems can have many modules and still be tractable is because the interconnections are contingent rather than logical. The designer has a greater degree of flexibility in determining which ones to attend to and which ones to ignore.

5.5. Incremental Development of Artifact

As interim design ideas or solutions are generated, they are retained, massaged and incrementally developed until they reach their final form. Very rarely are ideas or solutions forgotten or discarded. In other words, information about the state of the design, and associated knowledge brought into the design problem space, appears to increase in a monotonic fashion throughout the design process. This is one of the most robust findings in the literature on problem solving in design (Kant, & Newell, 1985; Ullman, et al., 1988).The duration of the incremental development process is, to a great extent, a function of the resources available.

There are a number of factors in the design task environment which would seem to favour a strategy of incremental development. First, the problems are large and, given the sequential nature of information processing, cannot be completed in a single processing cycle. Second, since there are few logical constraints on design problems and no right or wrong answers, there is little basis for giving up partial solutions and starting over from scratch. It makes more sense to continue to develop what already exists. Third, incremental development is compatible with the generation and evaluation of design components in multiple contexts that will eb discussed in the following section on control structure.

Incremental development does not occur in the nondesign protocols that we analyzed. While it is true that the nondesign problems were also too large to be completed in a single cognitive step, there is nonetheless a different character about the progression of knowledge states and problem states. More specifically, most of the search paths explored in finding solutions are wrong. When a particular search path is abandoned, much of the information associated with that path is also abandoned as the problem solver returns to a previous problem solving state and knowledge state and begins a new search path. The accumulation of knowledge relevant to a solution does not monotonically increase as the problem solver switches from one search path to another.

Incremental development in design, and its comparison to nondesign problem solving can be better understood by examining some particular protocol analyses and a more detailed discussion of the control processes operative in design problem solving. Both are discussed in the next section.

5.6. Control Structure

There are a number of issues which a control strategy for traversing design problem spaces needs to address. Among them are the following three:

1. Are the solution modules to be developed in isolation from each other, or is there to be interconnection between between the solutions?
2. Is the information to be processed sequentially or in parallel?
3. Are the solutions to be developed incrementally or appear completely formed?

We have already discussed results and assumptions regarding each of these issues: (a) the solution modules are interconnected to some degree, (b) the cognitive process is assumed to be sequential, and (c) the solutions are developed incrementally. A control strategy which accommodates and supports each of these facts is required.

Our data indicate that designers use a *limited commitment mode control strategy* (LCM control strategy). This strategy is closely related to Stefik's (1980) "least-commitment" control strategy. The basic feature of the LCM strategy is that, when working on a particular module, it does not require the designer to complete that module before beginning another. Instead, one has the option of putting any module on hold and attending to other related (or even unrelated) modules, and returning to the module on hold at a later time. This embedding can go several levels deep and one is not irrevocably committed to interim solutions. One always has the option of modifying partial design solutions at a later point. This, in effect, lets the designer take advantage of multiple problem-solving contexts in the generation and evaluation of design elements. To specify this more precisely, we need to (a) show that design elements are indeed considered in different contexts, and (b) trace out the actual control structure showing the LCM control strategy. In order to do (a) we first need a way to individuate design elements and design contexts.

5.6.1. Individuating Design Elements and Contexts

In the case of design elements there is a straightforward mapping between them and the modules and submodules which we identified earlier. Contexts are not as easy to individuate. They were identified using the following method: The protocol was divided into modules and submodules, and a unique number was assigned to each token occurrence of modules and submodules. Each numbered segment at the level of module or submodule constituted a different context. Whenever module or submodule types were instantiated, it was in a different context.

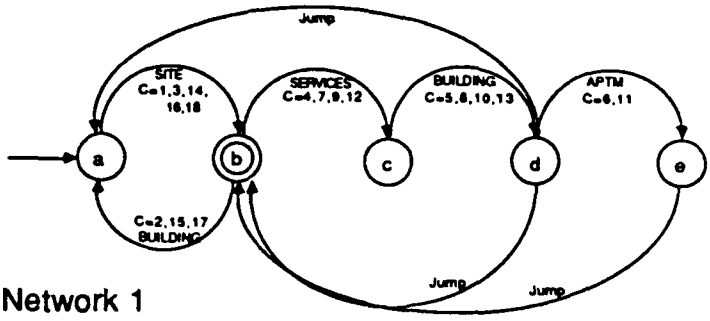
This notion of context seems to combine both temporal and content components, because whenever a uniquely numbered module or submodule was considered, it was at a unique time *t* and it was proceed and followed by a uniquely numbered sequence of modules or submodules (which means there was different information in the problem space). We have, for each of the subjects, a tracing of modules and submodules and the contexts in which they were considered. These tracings are best presented in conjunction with control strategies, to which we now turn our attention.

5.6.2. Control Strategies

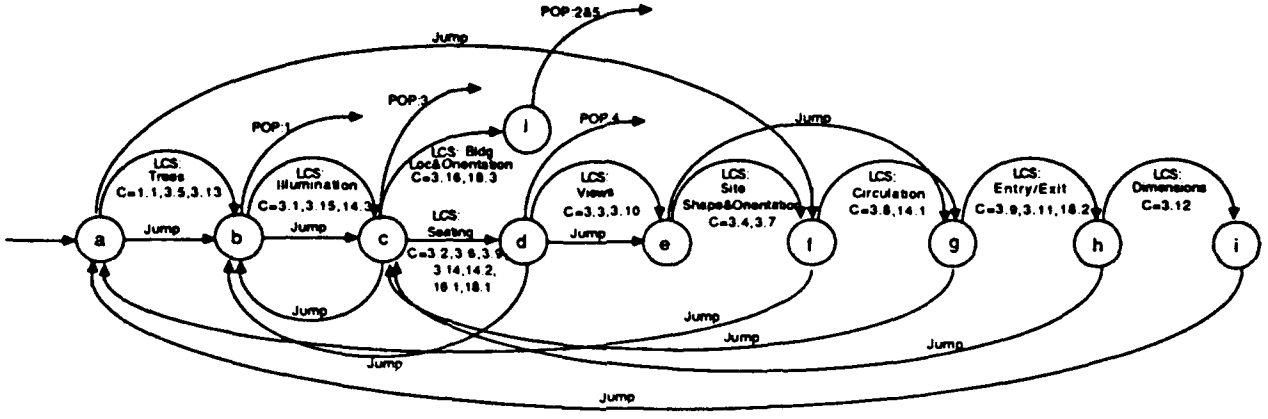
Control strategies can be enumerated at different levels. We used a three level hierarchical analyses which accounts for the protocols at the level of module, submodule, and individual statement types. As the module and submodule categories are task- and subject-specific, the control structure at these levels will also be task- and subject-specific. Since the categories at the level of individual statements are general across all the tasks and subjects, the control structure at this level will also generalize across tasks and subjects.

The actual formalism that we use to capture and display the control structure is a transition network. More specifically it will be a recursive transition network (RTN) (Winograd, 1983). These networks have been widely used in the computational linguistic field to recognize, parse, and generate natural language strings. We used them to (manually) recognize our protocols.

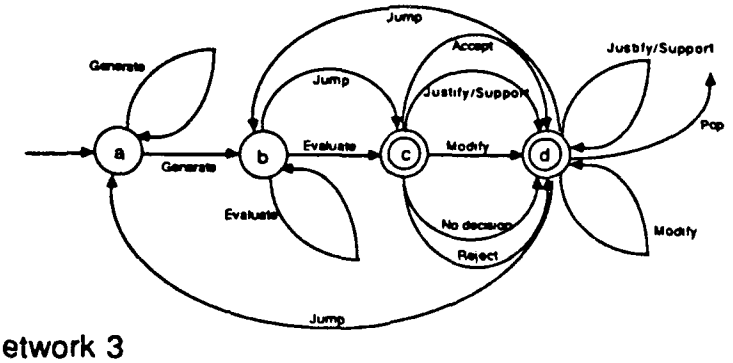
Some samples of control structure from subject S-A's protocol (one from each level) are presented in Networks 1, 2, and 3 (Figure 4). The reader is refered to Goel (1991) for a representation of the complete structure. The salient features of the networks are summarized below and the reader is invited to examine Figure 4 for details:



Network 1



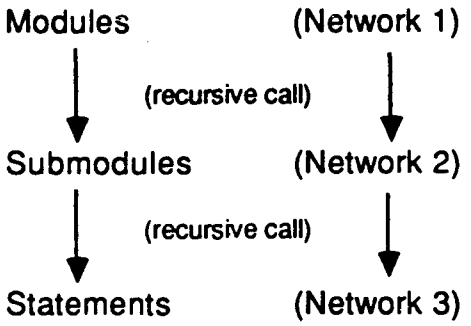
Network 2



Network 3

Figure 4. Sample of RTN networks that recognize Subject S-A's protocol. The "C" numbers specify the sequence (and context) in which the arcs are traversed. Network 1 recognizes the complete protocol at the level of the four modules. Network 2 recognizes the "site" submodule. Network 3 recognizes the local control structure of all the submodules.

- The control structure is naturally analyzed into three hierarchical levels:



- The first two levels are task-specific; the third is general across all three tasks.

- Within any of the levels, one does not get a consistent, steady, linear movement through the problem. Rather one gets repetitive, cyclical, flexible control structure at all three levels.

- The effect of this repetition and reiteration is that most modules and submodules are considered several times, in several different contexts. For example, in Network 1, the site module is considered 5 times, the building module 7 times, the services module 4 times, and the APTM module 4 times. Table 4 summarizes the multiple occurrences of submodules.

Table 4
Occurrences of submodules in multiple contexts for Subject S-A

Modules	Number of submodules considered	Percentage of submodules considered more than once	Range	
			High	Low
Site	9	89%	7	1
Building	18	28%	5	1
Services	4	25%	3	1
APTM	3	33%	2	1
Total	33	44%		

- The single Local Control Structure (LCS) Network can recognize all the modules in all the protocols. That is, it can do a flat or local statement by statement recognition of all the protocols.

- Most of the action in the protocols seems to be internal to major modules, at the level of submodules (Network 2). The top and bottom level networks (Networks 1, 3) are rather sparse and simple.

While the control strategy for the nondesign problem space look similar (in terms of the cyclical, repetitive revisitation of modules) to the LCM control strategy (see Goel, 1991) of the design problem space, much of the similarity is superficial. There is an important difference in the two cases. In the nondesign problem space most of the problem solving occurs internal to modules/episodes. There is little carryover from previous visits to a module or episode. In fact, Newell and Simon (1972) in their original analysis of these protocols claim that in returning to a former state, the subject is in fact returning to the previous knowledge state with respect to the problem. If he goes down the wrong path and returns to the previous state all that he knows is that the path he just explored does not lead to the goal state. He does not have an enriched understanding of the state he is returning to.

Table 5
Trace of the development of knowledge in
Subject NS6.1's repeated visits to Module 6

Visit #	Concluding Knowledge State	Development of Knowledge
1	G has to be an even number	•initial proposal
2	No letter in front of R	•connection with first visit unclear
3	G has to be either 1 or 2	•ignore/reverse previous conclusion
4	R has to be greater than 5	•connection to previous visits unclear
5	G is going to be 1	•maybe connected to visit 3
6	R = 9 G = 3	•may be connection to visit 4 •unconnected to any previous state

This can be demonstrated by tracing through the repeat visits to a module/episode and examining the state of knowledge at the end of each visit. Table 5 provides such a trace of subject

NS6.1's visits to the module column 6. The thing to note is the third, "development of knowledge," column. It indicates that there is not a close connection between the current visit to a module or episode and previous such visits.

In the design case, while problem solving does occur internal to modules, there is also considerable carryover and development of the module from visit to visit, as evidenced by the incremental development phenomenon, and further substantiated by the trace of Subject S-A's repeated visits to the submodule "seating" in Table 6. When the designer cycles back, it is not to the previous knowledge state, but rather to a previous topic instantiated in the current context.[4] This is indicative of some higher level control structure which we have not yet uncovered.

Table 6
Trace of the development of knowledge in Subject S-A's repeated visits to the "Seating" module

Visit No.	Concluding Knowledge State	Development of Knowledge
1	Keep seating below evergreens; don't disturb overall scheme	•initial proposal
2	Keep existing seating	•reaffirm original proposal
3	Incorporate seating elements as part of building structure; relaxation and view of playing field important	•builds upon first two visits
4	As per 3rd visit but decouple seating from building structure	•modify 3rd visit
5	Locate seating along boarders	•build upon 4th visit
6	Counter position seats so as to break up symmetry & not affect circulation	•build upon 5th visit

6. Summary and Conclusion

In this paper we have proposed a framework and method for the investigation of design as a cognitive process, and demonstrated some interesting results they yield. In particular, we started with some intuitions about the notion of generic design and put forward the design problem space hypothesis. To investigate the hypothesis we

[4]This is compatible with the view that design is an attempt at global optimization with finite resources.

(1) circumscribed design activity from non-design activity in an non-arbitrary and non-vacuous manner by
   (a) suggesting that design is a radial category exhibiting prototype effects,
   (b) examining some prototype cases and analyzing their task environments,
   (c) noting that not all task environments shared these features (i.e. not every task environment is a design task environment,
   (d) associating design activity with certain invariant characteristics of task environments,

(2) used the structure of the design task environment and a few well accepted constraints on the structure of information processing systems to generate hypothesis about twelve invariant characteristics of design problem spaces;

(3) analyzed data from both design and non-design tasks and found evidence for the postulated invariants. More particularly we presented data illustrating extensive problem structuring, distinct problem solving phases, reversing the direction of the transformation function, near decomposibility of design solutions, incremental development of design solutions, and a limited commitment mode control strategy. We were also able to explain why each of the invariants occur by appealing to the structure of the design task environment. Along the way we provided a detailed qualitative and quantitative characterization of design problem spaces.

(4) Equally importantly, we also presented evidence indicating that these invariants do not occur either at all, or at least not in the same form, in at some non-design problem spaces.

On the basis of this data and analyses we conclude that the notion of a *design problem space* is an interesting and explanatory theoretical construct and worthy of further study.

Author Notes

This paper is a condensed version of chapters 3 and 4 of the first author's dissertation (Goel, 1991). Funding for this research has been provided by the Office of Naval Research Cognitive Science Program, grant number N00014-88-K-0233 to Peter Pirolli, and a Myrtle L. Judkins Memorial Fellowship, a Canada Mortgage and Housing Corporation Scholarship, a Gale Fellowship, and a research internship at the System Sciences Lab at Xerox PARC and CSLI at Stanford University to Vinod Goel. We would like to thank Dan Berger, Kate Bielaczyc, Susan Newman, Mimi Recker, and Mike Sipusic for comments and discussions throughout this project. Correspondance should be directed to the first author.

References

Akin, O. (1979). An exploration of the design process. *Design methods and theories, 13* (3/4), 115-119.

Akin, O. (1986). *Psychology of architectural design.* London: Pion Limited.

Alberti, L. B. (1450/1988). *On the art of building in ten books.* Cambridge: MIT Press.

Alexander, C. (1964). *Notes on the synthesis of form.* Cambridge: Harvard University Press.

Alexander, C. (1965). A City is Not a Tree. *Architectural Forum*, Vol. 122, pp. 58-62.

Alexander, C., & Poyner, B. (1966). *The atoms of environmental structure.* London: Ministry of Public Building and Works.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Archer, L. B. (1969). The structure of the design process. In G. Broadbent&A. Ward (Ed.), *Design methods in architecture* New York: George Wittenborn, Inc.

Brown, D. C., & Chandrasekaran, B. (1989). *Design problem solving.* San Mateo, CA: Morgan Kaufmann.

Chandrasekaran, B. (1983). Towards a taxonomy of problem solving types. *AI Magazine, 4* , 9-17.

Chandrasekaran, B. (1987). Towards a functional architecture for intelligence based on generic information processing tasks. *Proceedings of the International Joint Conference on Artificial Intelligence* : Morgan Kaufman.

Cross, N. (1984). *Developments in design methodology.* New York: Wiley,

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31* , 1268-1287.

Darke, J. (1979). The primary generator and the design process. *Design Studies, 1* , 36-44.

Dertouzos, M. L., Lester, R. K., & Solow, R. M. (1989). *Made in America: Regaining the productive edge.* Cambridge, MA: MIT Press.

Dixon, J. R., & Duffy, M. R. (1990). The neglect of engineering design. *California Management Review.*

Eastman, C. M. (1969). On the analysis of intuitive design processes. In G. Moore (Ed.), *Emerging techniques in environmental design and planning*, Cambridge, MA: The MIT Press.

Eastman, C. M. (Ed.) (1975). *Spatial Synthesis in Computer-Aided Design.* N.Y.: Applied Science Publishers.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Gentner, D., & Gentner, D. R. (1983). Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner&A.L. Stevens (Ed.), *Mental Models.* Hillsdale, NJ: Lawrence Erlbaum.

Goel, V. (1991). Sketches of Thought: A Study of the Role of Sketching in Design Problem Solving and its Implications for the Computational Theory of Mind. Ph.D. Dissertation, University of California, Berkeley.

Goel, V., & Pirolli, P. (1989). Motivating the notion of generic design within information-processing theory: The design problem space. *AI Magazine, 10* , 19-36.

Greeno, J. G. (1978). Natures of Problem-Solving Abilities. In W. K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 5: Human Information Processing.* Hillsdale, NJ: Lawrence Erlbaum.

Greeno, J. G., Korpi, M., Jackson, D., & Michalchik, V. (1990). Proceedings of the Annual Conference of the Cognitive Science Society. (pp. 939-946): Lawrence Erlbaum.

Guindon, R. (1989). The process of knowledge discovery in system design. *Proceedings of the International Conference on Human Computer Interaction* : Elsevier Science.

Jeffries, R., Turner, A. D., Polson, P. G., & Atwood, M. E. (1981). The processes involved in designing software. In J.R. Anderson (Ed.), *Cognitive Skills and their acquisition* (pp. 255-283). Hillsdale, NJ: Lawrence Erlbaum.

Kant, E. (1985). Understanding and automating algorithm design. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1243-1253). Los Altos, California: Morgan Kaufman.

Kant, E., & Newell, A. (1985). Problem solving techniques for the design of algorithms. *Information processing and management, 20* (1-2), 97-118.

Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about the mind.* Chicago: The University of Chicago Press.

Larkin, J., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science, 11* , 65-99.

Liskov, B., & Guttag, J. (1986). *Abstraction and specification in program development.* Cambridge, MA: The MIT Press.

March, L. J. (1976). The logic of design and the question of value. In L.J. March (Ed.), *The architecture of form* Cambridge: Cambridge University Press.

Mellars, P. (1989). Technological changes across the Middle-Upper Paleolithic transition: Economic, social, and cognitive perspectives. In P. Mellars and C. Stringer (Eds.), *The human revolution: Behavioral and biological perspectives on the origins of modern hominids* (pp. 338-365). Edinburgh: Edinburgh University Press.

Mostow, J. (1985). Toward better models of the design process. *AI Magazine, 6* , 44-57.

Newell, A. (in press). Unified theories of cognition. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice Hall.

Perkins, D. N. (1986). *Knowledge as design*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M.W. Shelly&G.L. Bryan (Ed.), *Human judgements and optimality*. New York: Wiley & Sons.

Rittel, H., & Webber, M. (1973). Dilemmas in a general theory of planning. *Policy Sciences, 4* , 155-169.

Rittel, H. W. J., & Webber, M. M. (1974). Dilemmas in a general theory of planning. *DMG-DRS Journal, 8* (1), pp. 31-39.

Rosch, E. (1978). Principles of categorization. In E. Rosch & B.B. Lloyd (Ed.), *Cognition and categorization* (pp. 27-48). Hillsdale, NJ: Lawrence Erlbaum.

Schoenfeld, A. (1985). *Mathematical problem solving*. Orlando, FL: Academic Press.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society, volume 106* (pp. 467-482).

Simon, H. A. (1973). The structure of ill-structured problems. *Artificial Intelligence, 4* , 181-204.

Simon, H. A. (1973b). The Organization of Complex Systems. In H. H. Pattee (Ed.), *Hierarchy Theory*. N.Y.. G. Brazileer.

Simon, H. A. (1977). How Complex are Complex Systems? *Proceedings of the 1976 Biennial Meeting of the Philosophy of Science Association*, Vol. 2, pp. 507-522.

Simon, H. A. (1978). On the forms of mental representations. In C.W. Savage (Ed.), *Perception and cognition* Minneapolis: University of Minnesota Press.

Simon, H. A. (1981). *The sciences of the artificial (second edition)*. Cambridge, MA: MIT Press.

Spector, A., & Gifford, D. (1986). Case study: A computer science perspective on bridge design. *Communications of the ACM, 29* , 267-283.

Stefik, M. (1980). Planning and Meta-Planning (MOLGEN: Part 2). *Artificial Intelligence*, Vol. 14, No. 2.

Thomas, J. C. (1978). A design-interpretation analysis of natural English with applications to man-computer interaction. *International journal of man-machine studies, 10* , pp. 651-668.

Thomas, J. C., & Carroll, J. M. (1979). The psychological study of design. *Design studies, 1* (1), pp. 5-11.

Tong, C. and Franklin, P. (1989). Tuning a Knowledge Base of Refinement Rules to Create Good Circuit Designs. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Detroit, MI.

Ullman, D. G., Dietterich, T. G., & Stauffer, L. A. (1988). *A model of the mechanical design process based on empirical data* (Tech. Rep. DPRG-88-1).

Voss, J. F., Greene, T. R., Post, T. A., & Penner, B. C. (1983). Problem solving skill in the social sciences. In G.H. Bower (Ed.), *The psychology of learning and motivation: Advances in research theory* (pp. 165-213). New York: Academic Press.

Wade, J. (1977). *Architecture, problems and purposes: Architectural design as a basic problem-solving process.* New York: Wiley.

White, R. (1989). Production complexity and standardization in early Aurignacian bead and pendant manufacture: Evolutionary implications. In P. Mellars and C. Stringer (Eds.), *The human revolution: Behavioral and biological perspectives on the origins of modern hominids* (pp. 366-390). Edinburgh: Edinburgh University Press.

Winograd, T. (1983). *Language as a cognitive process, Volume 1: Syntax.* Reading, MA: Addison-Wesley.

Wynn, T. (1979). The intelligence of later Acheulian hominids. *Man*, 14, 371-391.

Wynn, T. (1981). The intelligence of Oldowan hominids. *Journal of Human Evolution*, 10, 529-541.

ATTACHMENT 3

AUGUST 1991 REPORT NO. DPS-5

The Structure of Ill-structured Problem Solving in Instructional Design

Peter Pirolli and Daniel Berger

University of California at Berkeley

Portions of this research were funded by the Cognitive Science Program, Office of Naval Research under contract N00014-88-K-0233, Contract Authority Identification Number NR4422550. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

The Structure of Ill-structured Problem Solving in Instructional Design

Peter Pirolli

and

Daniel Berger

University of California, Berkeley

August 1991

Abstract

In this paper, we present an analysis and a computer simulation model of a relatively ill-structured problem-solving task in instructional design. The primary motivations for this research include (a) extending information-processing analyses of well-structured problem solving to ill-structured tasks, and (b) uncovering the problem-solving and problem-structuring strategies employed in expert design. The simulation, implemented in Soar, of an expert instructional designer suggests that substantial activity is devoted to knowledge acquisition and integration as means for problem structuring. The heterogeneous mix of strategies uncovered by our simulations is compared to recent artificial intelligence approaches to design.

Introduction

Problem solving has been one of the dominant fields of research in the study of human information processing over the past three decades (Greeno & Simon, 1988) and such research has left its signature on several major theories of human cognition (Anderson, 1983; Newell, 1990). It is notable, however, that the bulk of this research has focused on problems that are relatively *well-defined* or *well-structured* (Reitman, 1964; Simon, 1973). In other words, the sorts of tasks that have typically been studied involve problem situations, goals, and problem-solving actions that require little embellishment on the part of problem solvers. Consequently, theories of problem solving have been largely shaped by the empirical phenomena of well-structured problems, and it is reasonable to question how well the theories extend to relatively *ill-structured* problems, which are far more characteristic of the types of dilemmas faced by individuals in their everyday endeavors (Reitman, 1965).

In this paper, we present an analysis and a computer simulation model of a relatively ill-structured problem-solving task in instructional design. In part, our aim is to examine how current conceptions of problem solving extend to ill-structured problem solving. As part of our research strategy, we have used the Soar framework (Newell, 1990) to formalize our simulation model and to guide our empirical analyses. Soar is quite explicitly a computational cognitive modelling formalism grounded in concepts and assumptions that have emerged from research on relatively well-structured problem solving. By formalizing our analysis in Soar, we not only test the sufficiency of our particular account for the data at hand, we also test how well Soar and its underlying conceptions of problem solving extend to ill-structured tasks. A clear precedent for this kind of analysis is Simon's (1973) discussion of how GPS--a direct ancestor of Soar--might solve ill-structured problems in areas such as architecture and ship design. Our research could be construed as a modest attempt to test the notion that "the problem solving mechanisms that have shown themselves to be efficacious for handling large, albeit *apparently* well structured domains should be extendable to ill structured domains without any need for introducing qualitatively new components" (Simon, 1973, p. 197; see also Reitman, 1965).

The work presented here also arises out of our interests in studying the structure of design as a particular kind of problem solving. Expertise and creativity in design has traditionally been highly valued, yet we know very little about it. Creative and effective design involves skillful generation, manipulation, and monitoring of strategies and tactics, abilities to generate and reason with representations of the world, the analysis and integration of complex information, and many other phenomena of perennial interest to cognitive scientists. Moreover, we assume (Goel & Pirolli, 1989) that ill-structured problem solving across design disciplines shares many features, and yet differs substantially in character from other kinds of problem tasks such a game-playing, puzzle solving, solving word problems, or diagnosis. If this assumption is warranted, then studies of design tasks should yield information about a broad "natural kind" of human activity that is of both practical and scientific utility.

An additional aim of this research is examine the variety of control strategies that occur in human design. Workers in artificial intelligence (AI) have proposed a number of general control regimes as basic approaches to design problem solving (Brown & Chandrasekaran, 1989; Chandrasekaran, 1987; Chandrasekaran, 1990; Kant, 1985; Maher, 1990; Mostow, 1985; Mostow, 1989). Cognitive models of human design can

both draw upon and inform this research by indicating the mix of control regimes and problem representations employed for design subtasks by human experts.

Ill-structured Problem Solving

In Newell and Simon's (Newell & Simon, 1972) highly influential theory, problem solving is characterized as a process of *search* in *problem spaces*. A problem space is a formal characterization of the structure of processing that results from an interaction of the human information-processing system with some *task environment*. The structure of a problem space is defined in terms of the states of problem solving, operators that move the problem solving from one state to another, and the goals or evaluation functions of problem solving. The task environment is the portion of the external environment relevant to the problem-solving task.

Reitman (1964, 1965) presented seminal discussions of well-structured and ill-structured problem solving in terms consistent with this classical framework of human problem solving. Reitman (1964) proposed that problems were ill-defined to the extent that problem states, goals, or transformation functions lacked information, were ambiguous, or included open constraints. He also suggested that such ill-defined problems, which people take on "with nothing remotely resembling a statement of the necessary and sufficient conditions for a solution" (Reitman, 1965, p. 148), are the focus of largest percentage of human energies, yield larger interindividual variability in solutions, and produce less agreement on the acceptability of solutions, when compared to well-defined problems.

Inspired by Reitman's work, Simon (1973) argued that there was nothing intrinsic about a problem, per se, that made it ill- or well-structured, but rather that such properties could only be determined by examining the relationship between the problem solver, its available knowledge, and the problem to be solved. Furthermore, Simon (1973) argued that information-processing accounts that had been applied to the study of relatively well-structured problems were adequate to deal with relatively ill-structured problems.

Simon (1973) proposed that processing in ill-structured situations entails substantial integration and manipulation of knowledge that ultimately provide enough structure to a problem space to yield an acceptable solution. The sources of such knowledge are the long-term memory of the problem solver, and knowledge relevant to the task integrated from external sources, including the problem description. The structure of processing and the ultimate solution are, in large part, a reflection of the particular knowledge used to structure the problem space.

Instructional Design

In this study, we have focused on the processes involved in instructional design tasks by professional instructional designers. In related research (Goel & Pirolli, 1989), we have developed an abstract characterization of the features of tasks that are prototypical of design disciplines (e.g., architecture, bridge design, or mechanical engineering), and professional instructional design fits this characterization rather well. On this basis, we expect that many of our findings about instructional design will generalize to other design tasks.

Typical instructional design tasks for a professional might, for example, involve developing training materials about a new product for the technicians who will have to service the equipment, or materials to train sales personnel on appropriate strategies for

dealing with potential customers. Although many constraints may be fixed (e.g., time, money, and other resources available), many may still be open or undefined. For example, a designer may have the option to decide upon the medium for the instruction (e.g., videodisk versus texts), or perhaps many attributes of the target students may be unknown. Typically, the designs take weeks to months and, once completed, the instruction is implemented away from the designers, either because it is stand-alone instruction, or because it is taught by a separate cadre of teachers or trainers. Although instructional designers will sometimes perform what are called *formative evaluations* to determine the effectiveness of their designs during the design process, true feedback on a design's effectiveness occurs through *summative evaluations* of the end-products of the design process.

Like many other design disciplines, instructional design has developed a large body of prescriptive theory and methodology. A useful characterization of prescriptive instructional theory (Reigeluth, 1983) is that instructional design theories attempt to specify the space of instructional *situations*, the space of instructional *methods*, and to develop statements, called *principles*, that link these spaces. The analysis of instructional situations is taken to broadly include the effects of instructional methods, usually called instructional *outcomes*, and the *conditions* that affect the outcomes and use of those methods. Such conditions include the subject-matter, the instructional setting, properties of the targeted learners, and the nature of the learning task.

Principles of instruction are taken to be those statements that characterize elementary building blocks for instructional methods. *Descriptive* principles are scientific statements about the effects of a particular method under given conditions. *Prescriptive* principles are the kind of statement used in design to identify the optimal method to use in a given situation. Comprehensive instructional theories are intended to provide the knowledge base for solving the problems of instruction by providing the rules and constraints for forming efficient and effective instructional interactions.

Insert Figure 1 about here

Prescriptive instructional design methodologies specify how design should be carried to yield effective design solutions. One popular method (Mager, 1988) for the design of industrial training, is presented in Figure 1. This particular specification of instructional design consists of four phases: *analysis, development, implementation, and improvement.* The analysis phase can be thought of as background research that will be relevant to structuring and solving the design problem. Mager (1988) emphasizes that the main purpose of this phase is to determine the goals of the instruction. The development phase is defined as procedures which include "the drafting of measuring instruments, as well as development and tryout of the instruction itself" (Mager, 1988, Volume 1, p. 15) As would be expected, the implementation means the delivery of the instructional materials to the students. The improvement phase can be seen as the feedback of the summative evaluations into the design process for further refinement of the instruction.

The Protocol Database

Our protocol database was obtained from nine professional instructional designers for a large multinational corporation whose main product is office systems. The subjects were volunteers who had responded to a request for participation in the study by their managers. All subjects were familiar with the Xerox Star system and its Viewpoint

document preparation environment, which were the subject-matter for the instructional design task. Four women and five men with varying ages, backgrounds and expertise participated in the design experiment.

In part, our selection of professional designers and the design task was shaped by an earlier pilot study, in which we had collected protocols from two professors, a lecturer, and a teaching assistant, from the University of California at Berkeley, who were all experienced in teaching statistics. These pilot subjects had been asked to design a lesson on the law of large numbers to be delivered to psychology freshmen by a teaching assistant familiar with statistics. We noted that (a) specifying lessons in advance did not appear to be a familiar activity for the pilot subjects, (b) the subjects did not produce specifications of uniform detail or concreteness, nor did they seem to have a sense for what should constitute a specification, (c) the subjects appeared to be hampered by the lack of resource material on the topic,[1] and (d) subjects posed many questions about constraints and resources for the design during the design sessions. Consequently, we contacted professional designers who were familiar with specifying instruction in advance and producing assorted specification documents, which are highly stylized to include particular kinds of information about an instructional design. In addition, we provided technical documentation on the topic that was the focus of the instruction (i.e., manuals for the word processor), and we developed a set of scripted answers to questions we thought might arise during the design sessions.

Procedure

Subjects were studied individually as they performed the design task. At the beginning of the design session subjects were presented with task instructions and a short design brief. The task instructions informed subjects that they had three hours to complete the design, that the session would be video taped, and that they were requested to "think aloud" during the experiment. At a convenient point during the session, the subject was given the opportunity to take a short break.

The design brief described a fictional company (Acme Umbrella) that was about to automate its administrative facilities by replacing secretaries' typewriters with Xerox Star computers and its Viewpoint documentation preparation environment. The brief also stated that Acme had observed that the manuals accompanying the computer would not provide adequate training for the secretaries. The design brief stipulated that the subject was being hired freelance to develop an appropriate training package for Acme's secretaries.

The training package was to be made up of six one-hour lessons of stand-alone instruction. Other constraints placed on the training package in the design brief were varied across subjects along three dimensions with two cells for each dimension:

- *The instructional delivery medium.* The stand-alone instruction requested was either (a) text-based or (b) interactive video-disk
- *The background knowledge of the secretaries.* Subjects were told that the secretaries (a) knew just how to use an electric typewriter or (b) had some experience with a word processing system.

---

[1] In fact, all four pilot subjects produced a lesson on the *central limit theorem* rather than the law of large numbers, probably because the former is a more central topic for psychology students.

- *The form of the specifications expected at the end of the design session.* Subjects were asked either (a) to organize all the lessons and design the first hour in as much detail as possible (depth-first design) or (b) to design the complete package in as much detail as possible (breadth-first design).

These dimensions and cells create eight variants of the design brief. Assignment of subjects to design briefs was not completely random, since four of the designers who had some experience with designing interactive video disk were assigned to work on design briefs that requested interactive video disk instruction.

There were two experimenters present during the design sessions, and subjects were told that one of the experimenters could be treated as a representative of Acme Umbrella and would answer any questions about the design brief. Manuals for the Viewpoint environment were on hand and available for the designer to use at any point. Subjects were given an unlimited amount of blank paper to create their design specifications. Subjects worked at a desk and two VHS camcorders were used to record the design sessions. The first camcorder was aimed at the subjects' work area on the desk to record all writing activity whereas the second used a wider-angle view to capture a broader image of the designers. These videotapes and the designers' notes served as the raw data for subsequent protocol transcription and coding.

Protocol Coding Scheme

The video tapes and writings of all nine designers were subjected to a first pass of qualitative analysis in which we identified and compared the kinds of specifications produced, a general characterization of the content of the specifications, and a general characterization of the subtasks that the designers had performed. Detailed protocol analyses were conducted on data from three designers. These three subjects were selected because they were highly verbal, and because collectively their protocols covered a wide variety of the phenomena we had noted among all nine designers.

The first of these three designers, Subject W, had one year of college and was working on a degree. He had about 10 years of work in a variety of technical professions, had been a trainer of technicians for about 4.5 years, and worked as a designer, analyst, and writer of instruction for eight years. The second designer, Subject M, had completed undergraduate degrees in psychology, with a focus on human factors and computer-assisted instruction. She had worked as a technician for three years, a trainer of technicians for seven years, and worked in instructional design for 2.5 years. The third designer, Subject C, worked as manager for instructional design projects. He had an undergraduate degree in psychology, a masters in education, and had completed about one third of the requirements for a Ph.D. in education. Subject C was also an adjunct faculty at a local institute of technology and had taught courses on topics related to instructional design. He had worked as a technician for four years and as an instructional designer for 19 years.who were asked to design instruction for a text editor following a procedure described in detail below.

The verbalizations recorded on the videotapes were transcribed and cross-referenced with the notes written and drawn by subjects. The transcribed protocols were segmented into utterances and actions following previous outlines of verbal protocol analysis (Ericsson & Simon, 1984). Utterances were individuated into phrases conveying single ideas either based on content cues, such as shifts in topic or new points being made about a topic, or by noncontent cues, such as pauses, phrase and sentence boundaries, and

the making and breaking of contact between pen and paper. Our coded transcripts also contained codings of all reading, writing, and erasing actions performed by subjects.

We coded each utterance along three primary dimensions: (a) the kind of *operation* being performed, (b) the *information type* being operated upon, and (c) the inferred *source* of the information. Each coding of an information type also included a set of arguments associated with the particular type that were used to specify the particular content of the utterance. A summary of the protocol coding scheme is presented in the Appendix, and a more detailed presentation of the scheme, along with the coded protocols of subjects discussed in this paper are available from the authors.

Analysis and Simulation of an Instructional Design

Most of our efforts have focused on the analysis and development of a simulation model that addresses the protocol of Subject W. This subject was selected for detailed attention because (a) he was more verbal and articulate than average among our subjects, (b) he proceeded farther in the task than other subjects, completing a draft of the actual text for a lesson, and (c) he attended to a broader range of design and analysis subtasks than most of our subjects, although the subtasks he worked on were exhibited in the protocols of other subjects.

An Overview of Subject W's Design Protocol

The design brief presented to Subject W stated that his task was to design instruction for a fictional company (the Acme Umbrella Company). The brief stated that Acme was automating its offices, and part of this process involved the replacement of typewriters with Xerox Star computers and its Viewpoint documentation preparation environment. Subject W's task was to design six hours of stand-alone text, preferably divided into one hour slots, to train Acme's secretaries on the use of Viewpoint. Each secretary was expected to work on one lesson per day over two weeks. At the end of the period, each student was expected to be a competent user of Viewpoint. The brief noted that these secretaries were "familiar with electronic word processing" and that the secretaries would have access to the Star workstations during training. Subject W was asked to "organize the lessons and design in detail as much of the 1st hour as possible." In addition, Subject W was asked to produce a design that would "enable the client to extrapolate what the complete package will look like," to "convince the client it will meet his instructional objectives," and to "take full advantage of the expressive potential of the medium."

Figure 2 presents a summary of Subject W's writing output and protocol statements up to the point at which he began writing his draft of the instructional text. The section of protocol summarized in Figure occupied about 1.5 hours. The abscissa in Figure 2 marks the sequence of coded protocol statements (943 total) uttered by Subject W over time, the ordinate marks different information type statements. Figure 2 is also segmented vertically, with segments labelled A to G corresponding to different kinds of written output produced by Subject W.

Insert Figures 2, 3, and 4 about here

To provide a general sense of Subject W's design process, it is useful to first look at the things that he wrote. After some preliminary discussion of the design task (the first,

unlabelled, segment in Figure 2), Subject W wrote 11 pages containing notes which can be grouped into the following seven segments (labels in this list correspond to the labels in Figures 2, 3, and 4):

(A) Subject W wrote one page of notes labelled "customer expectations." The notes included points about the length and structure of the instruction, the main objectives of the course, and attributes of the secretaries.

(B) Next, Subject W wrote $1\frac{1}{3}$ pages of notes labelled "order of training" and sublabelled "task listing." The notes were a hierarchical (i.e., nested) list of tasks and concepts that are related to creating and editing documents. Much of the list was produced while skimming through the Viewpoint manuals.

(C) Subject W then wrote about $\frac{2}{3}$ of a page of notes labelled "session 1" and sublabelled "overview course." The notes concerned an overview of the purpose and expectations for the course that was to form the initial segment of the first lesson.

(D) Subject W began a more detailed set of notes about the course overview for the first lesson on a separate page labelled "overview." The overview had two main subsections labelled (a) "why star," which contained notes about the motivation for using the Star system, and (b) "what will be trained," which contained notes on discussing the structure of the course and the topics of the six sessions. This segment ended with about $\frac{3}{4}$ of a page complete, after Subject W had written "sessions to include" as a subsection of "what will be trained."

(E) Subject W then wrote $\frac{3}{4}$ of a page of notes labelled "prioritization of 6 sessions" in which he organized the topics that needed to be taught in the course. Much of this was generated by reexamining the task analysis produced in Segment B and the objectives developed in Segment A.

(F) Subject W then returned to the notes he had started in Segment D, completing the page of notes and about $\frac{1}{4}$ of another page with an enumeration of the six sessions and their topics

(G) Subject W then wrote four pages of notes labelled "outline of session # 1 familiarization" that provided a detailed list of the topics and subtopics to be covered in the first session.

After completing these notes, the subject took a short break and worked on the draft of the first lesson.

General examination of Figure 2 reveals that within each segment, Subject W had focussed on some particular subset of content, but occasional references were still made to other aspects of content. The initial segment of Figure 2 indicates that Subject W largely discussed available knowledge sources (labelled "K. Sources" in Figure 2; e.g., the

Viewpoint manual), and features of the experimental task (labelled "Exp. Task"). Segment A is comprised largely of statements about the objectives, instructional constraints, student attributes, and the conditions (at Acme) before, during, and after the proposed instruction. Segment B largely concerned the content of the proposed course. Sections C, D, F, and G largely concerned the specific instructional transactions that would take place in the the course. Section E was largely devoted to discussion of the sessions that would constitute the course.

Figures 3 and 4, which simply replot portions of Figure 2, indicate our inferences about the source of the knowledge underlying design statements. Figure 3 plots only protocol statements in which the content of the statement was identified as originating from external sources: the experimenter, the design brief, or the ViewPoint manual. Figure 4 shows protocol statements in which the content was inferred as originating from the subject (self), or was an elaboration of previous content, a derivation from previously stated content, or a review of previous content. Comparison of Figures 3 and 4 illustrates the general flow of knowledge throughout the problem solving. At the beginning of the task, knowledge from external sources (Figure 3) dominated the content of the protocol, the early to intermediate stages of the protocol have higher rates of self-generated content (Figure 4), and the later portions of the protocol are dominated by elaborations, derivations, and reviews of content (Figure 4).

Overview of the Simulation

Overview of Problem Solving in Soar

The simulation of Subject W is implemented in Soar, which has been proposed by Newell (1990) as a universal theory of cognition. Soar performs tasks by problem space search using *universal subgoaling*, and uses a base learning mechanism called *chunking*. Problem spaces comprise sets of states, with operators that move from state to state, or that augment a current problem solving context. The achievement of desired states within a problem space constitutes task accomplishment. Knowledge is used to define and select problem spaces, to guide progress towards desired states, and to select and apply operators All long-term knowledge is stored in a content-addressable memory, represented as a production system.

The processes generated in performing a task are focused by *goal contexts* that are stored on a *goal stack*. A goal context is basically a four-slot frame with slots for a goal, a problem space, a state, and an operator. The conditions of productions test for patterns in the goal stack, which is stored in a working memory, and the production actions can add new information to working memory or implement basic operators. Some complex operators may involve significant search for their implementation, and thus may be implemented as tasks themselves in appropriate problem spaces. In addition, the process of operator selection itself may be implemented as a complex task in problem spaces (e.g., achieving the computation of search heuristics). Thus, the full power of problem space search can be applied to any subcomponent of a task.

Soar operates by iterating through *decision cycles*. Each decision cycle includes an *elaboration phase* and a *decision phase*. In the elaboration phase, knowledge relevant to the current task context is retrieved by the parallel firing of productions. The actions of some productions will deposit *preferences* about possible actions to take. Preferences are basically assertions stated in a domain-independent format concerning the selection, rejection, or comparison of problem spaces, states and operators for goal contexts on the

goal stack.[2] That is, preferences are statements about how the slots in goal contexts should be changed, and thus determine changes in problem spaces, states, and operators. For example, given a particular goal context containing a specific problem space, state, and operator, a production may assert that a new state is "best" for that goal context. Such a preference would achieve the application of the operator by moving from the current state to the new state.

The end of the elaboration phase occurs when all relevant productions have been fired (i.e., quiescence is achieved). At this point, the decision phase is invoked and the set of preferences are examined to determine a unique action to take (using a task-invariant comparison scheme). Frequently, the available knowledge is insufficient or conflicting in determining an action. Such a situation gives rise to *impasses*, which automatically generates a subgoal. Thus, new goal contexts are added to the goal stack solely through the generation of impasses. In these subgoals, knowledge can be brought to bear in selecting new problem spaces and in guiding search in order to overcome the impasse. When an impasse is overcome, the subgoal disappears from the goal stack. Chunking summarizes the problem solving (by creating new productions) that lead to the resolution of an impasse by tracing through dependencies from the working memory elements that produced the solution back to the working memory elements that were available in the context that triggered the impasse. In our simulation of Subject W, the Soar chunking mechanism was inactive, since we did not have many converging measures of learning for Subject W.

Specification of the Instructional Design Task

We represent the desired state (the goal) of problem solving in design as a set of constraints that must be satisfied. More concretely, the desired state requires that there be a specification of an artifact that transforms the current situation into a desired situation within the constraints imposed on the design specification and the design process. For instructional design this desired state is represented as a schematic structure of the following form (in pseudo-Soar notation):

(goal <g> ^desired <d>)

(instructional-design <d>
^current-situation <c_1> <c_2> ... <c_n>
^target-population <t_1> <t_2> ... <t_n>
^desired-situation <ds_1> <ds_2> ... <ds_n>
^delivery-constraint <dc_1> <dc_2> ... <dc_n>
^specification-constraint <sc_1> <sc_2> ... <sc_n>
^instructional-specification <i>)

where *goal* and *instructional-design* are objects represented as lists containing a unique identifier (in this case, <g> and <d>), and sets of attribute-value pairs. Attributes (or slots) are prefixed with the up-arrow, ^, and are followed by one or more values. The desired state in our simulation is structured as shown above, and contains information that is present in Subject W's design brief. The *current-situation* slot contains information about the pre-training situation of the imaginary Acme Umbrella Company. The *target-population* slot contains information about the secretaries of that company. The *desired-situation* contains information about what the secretaries should know after the training (i.e., they have to learn a word processor). The *delivery-constraint* slot contains information about

[2]The simulation is implemented in Soar 4.5. In a the more recent Soar 5.0 version, preferences can be asserted for any worming memory augmentation.

how the training is to be delivered. The *specification-constraint* slot contains information about the length of time available for design and general constraints on what the final specifications should contain. The *instructional-specification* slot contains no information, and it must be filled by a specification that satisfies the other constraints.

General Structure of the Instructional Design Task

Much of the problem solving in our simulation occurs in *specification* problem spaces. The basic problem solving operations are (a) collecting information (from the world or LTM), (b) decomposing a specification into subspecifications, (c) specifying the subspecifications, and (d) registering when a specification is done. Figure 5 presents an overview of the specification problem spaces in our simulation. The triangles in Figure 5 represent problem spaces, the circles represent states within those spaces, and the solid arrows indicate operators that change state information. The dotted lines indicate that impasses in applying an operator may result in a switch to problem solving in another problem space.

Insert Figure 5 about here

The specification problem space is selected in our simulation when some aspect of the design must be specified. A *decompose-specification* operator may apply and indicate how the specification may be decomposed into subcomponents which, if specified, would comprise the whole specification. For instance, the task of specifying the complete instructional design is decomposed by an operator into subspecifications consisting of (a) the customer expectations, (b) the target population, (c) the task analysis, and (d) the lesson plans. We assume that this particular decomposition is a piece of general planning knowledge held by Subject W (it is also consistent with (Mager, 1988)). These subspecifications are processed by attacking each in their own specification problem space, where additional decompositions may take place. If a decompose-specification has applied in a specification problem space, an *assemble-specification* operator recognizes when all the subspecifications have been completed and acts as a goal test that terminates processing in a particular specification problem space.

If no decomposition is retrieved from LTM, then by default a *get-information* operator is selected. This causes the simulation to go into a problem space in which further information is collected and evaluated with respect to the current specification task. For instance, in specifying the task analysis, the simulation makes use of the table of contents for the Viewpoint manual. The table of contents is available as a structured text that Soar must access through its general i/o interface. Individual text headings are read into working memory which, in turn, cause the retrieval of LTM information about the word processor and its use. The simulation uses the external table of contents to develop a plan-like representation of the tasks involved in text editing, and the relevant features of the text editor that need to be acquired in order to create and edit documents.

In general, the task-general planning knowledge is captured by decompose-specification operators. These include plans for specifying an instructional design (discussed above), plans for specifying customer expectations (e.g., objectives), and the general structure of individual lessons. Problem-specific aspects of the design, such as the specific task analysis and specific target population analysis, are carried out by the get-information operator and its associated problem spaces. These latter problem spaces serve to construct and evaluate mental models of the instructional situation and artifact. Roughly,

the design process involves the development of a mental model of the instructional situation, desired outcomes, and the instruction itself, and the development of a set of specifications that capture the essential features of this model as a kind of external blueprint.

Interaction with the Outside World

The Soar architecture comes with a facility, Soar I/O, that allows for interaction with the "outside world." All interactions with the outside world take place through the state associated with the top-level (or base) problem space in Soar. Information added as certain augmentations of the top-level state automatically generate output, and similarly, inputs from the outside world are deposited as specific augmentations of the top-level state. The information in the top-level state is accessible from any subgoal context.

In the simulation of Subject W, Soar I/O is used during three different situations. The first is to simulate the subject's oral interaction with the experimenter (acting mainly as the "Acme representative"). The second use of Soar I/O is to simulate Subject W's input of information from the manual and other external documents (including notes written earlier by Subject W). The third use of Soar I/O is to simulate the writing produced by Subject W.

Almost all of Subject W's interactions with the experimenter were questions that occurred during the opening period of the protocol. The inquiries can be classified as two types: verifications and clarifications. The former were questions requiring "yes" or "no" answers, while the latter were requests for more information. Soar I/O is used to simulate these interactions. The responses to the questions are simply stored in lists external to Soar and the content and sequential ordering of the lists correspond to the answers provided in the experimental session. The hierarchical table of contents of the Viewpoint manual is represented in a hierarchically organized tree data structure external to Soar. The output of the simulation is also written to hierarchically organized data structures.

A Comparison of the Simulation and the Protocol

We now turn to a detailed discussion of the protocol of Subject W and our Soar simulation of that protocol. Our simulation addresses the segment of Subject W's protocol that occurs after he had read the instructions for the experiment and the design brief and had asked a few general questions about the experimental task (e.g., about using pencil and paper; clarifying that an experimenter would be treated as a client). The segment addressed by the simulation ends at the point where Subject W began actually writing a draft of the instructional text he had designed.

Initialization of the Design Task and the Generation of the Top-level Specification Task

We present the first few cycles of operation of our simulation in substantial detail. In addition to some interesting features of the protocol that are being simulated, the detailed presentation is also intended to provide the reader with a more concrete view of the operation of the Soar simulation. In particular, the detailed presentation is intended to provide some sense of how problem spaces, states, and operators are created, selected, and terminated, how impasses occur, how processing in subgoal contexts can be used to implement operators that occur in supergoal contexts, and how input and output are carried out. In later sections, our presentation is less detailed.

The top-level goal context in our simulation is initialized as being in a *design-instruction* problem space with an initial state that contains elements that encode the fact that there is a client present, a notepad for output, and an available Viewpoint manual. As suggested above, the top-level goal context also includes a specification of the desired state for the task that summarizes the material that is in Subject W's design brief. More specifically, the information for our simulation includes:

- *Current situation.* The Acme company is buying Star computers and it is buying the Viewpoint system.
- *Target population.* The student group consists of secretaries who know word processing and MacIntosh computers. The secretaries do not know how to perform word processing in Viewpoint.
- *Desired situation.* The secretaries know how to perform word processing in Viewpoint.
- *Delivery constraints.* The instruction must take six hours, be carried out over two weeks, and be stand-alone text. There must be six one-hour lessons.

Also, as suggested above, a slot for an instructional specification is present but unfilled.

There is an interesting interaction that takes place at the beginning of Subject W's protocol (E refers to the experimenter and S is Subject W):

E: We have approximately three hours. If at any point during that period you want to divide the work up, if you want to take a five or ten minute break, feel free to suggest it.

S: Ok. [pause] And we are understanding by this that the, that this is the right system for them?

E: Yes.

S: We are not going to go in with IBM PCs or anything else?

E: Right. That's the system.

S: Ok.

At the beginning of this segment, the experimenter was finishing up some discussion of the general task constraints. Subject W then appeared to begin the design task by first checking that there was no chance that the clients would change their minds about buying the Star system after the instructional design was completed. This expectation failure check is highly reminiscent of work on case-based planning (***ref***), in which specific cases of planning are retrieved from memory and used to identify possible ways (and reasons) that a current plan may fail. From the protocol data it is impossible to identify whether or not Subject W was thinking about a specific past experience or some generalization of past experiences, but it does seem that past experiences are generating warnings for the current design plan.

The trace of our simulation for this early segment of protocol is summarized in Figure 6. Given the top-level goal context specified above, the simulation proposes two operators: (a) a *specify* operator which will generate the instructional specification, and (b)

a *check-expectation-failure* operator that insures that the clients will not change their minds. More specifically, the following production (in pseudo-English) proposes the specify operator:

P1: design-instruction*specify-instruction*acceptable
IF in the goal context <g>
the problem space <p> is design-instruction
the desired state <d> includes instructional-specification <i>
the current state <s> does not have an instructional-specification
THEN create a specify operator <q>
whose result will be <i> and an object of type instructional-specification
the preference for <q> is acceptable in context <g>, problem space <p>, and state <s>

Items enclosed in angle brackets, such as <g>, are variables. Production P1 detects that the current state does not contain the desired instructional specification and creates an operator to generate the specification. The operator is assigned an *acceptable* preference for the current goal, problem space and state, which means that the operator can be selected for application.

Insert Figure 6 about here

In parallel, another production proposes the check-expectation-failure operator

P2: design-instruction*check-expectation-failure*change-system*acceptable
IF in the goal context <g>
the problem space <p> is design-instruction
the current state is <s>
the desired state <d> includes a current situation which includes
the buying <b> of an office computer system
THEN create a check-expectation-failure operator <q> to check expectation <b>
the preference for <q> is acceptable in context <g>, problem space <p>, and state <s>

Production P2 detects that the instructional design problem is one involving the purchase of an office computer system and it generates an operator to check that that expectation will hold.

The parallel application of productions P1 and P2 at this point in the simulation has created two operators, both of which can apply to the current context. In the next cycle of the same elaboration phase production P3 is executed, which generates a best preference for the check-expectation-failure operator in the current context:

P3: design-instruction*check-expectation-failure*best
IF in the goal context <g>
the problem space <p> is design instruction
the current state is <s>
there is an acceptable preference
for a check-expectation-failure operator <q>
THEN the preference for <q> is best in context <g>, problem space <p>, and state <s>

After P3 is executed, quiescence is achieved in the elaboration phase, and the Soar decision phase is executed. The preference processing scheme in Soar identifies the check-expectation-failure operator as the unambiguously best operator, and the operator is installed in the current goal context. It is important to note that the specify operator and its acceptability in the current goal context are not removed by the selection of the check-expectation-failure operator.

During the next elaboration phase, no knowledge is retrieved to implement the check-expectation-failure operator. The goal context therefore remains unchanged during the following decision phase and consequently a *no-change impasse* occurs. This causes a new subgoal context to be automatically generated by Soar, and pointers to information about the source of the impasse are also automatically generated by the Soar architecture. Most importantly, there are pointers from the subgoal context to information that identifies the operator that caused the no-change impasse.

In the elaboration phase following the generation of the new subgoal context, the following production applies to create a problem space and initial state which will be used to implement the check-expectation-failure operator:

P4: check-expectation-failure*space&state
IF in the goal context <g> the problem space is undecided
the context occurred because of a no-change impasse on operator <q>
<q> is a check-expectation-failure operator for expectation <e>
THEN create a check-expectation-failure problem space <p>
the preference for <p> is acceptable in context <g>
create new state <s> containing expectation <e>
the preference for <s> is acceptable in context <g>

This problem space and state then gets installed in the goal context during the following decision cycles. In a subsequent elaboration phase, production P5 recognizes that there is a client who can be asked to evaluate the expectation:

P5: check-expectation-failure*ask-client*acceptable
IF in the goal context <g>
the problem space <p> is check-expectation-failure
the current state <s> contains expectation <e>
the top state contains a client <c>
THEN create an ask operator <q> to ask <c> for an evaluation of <e>
the preference for <q> is acceptable in context <g>, problem space <p>,
and state <s>

Once the ask operator is selected, a production executes which augments the top state with a query to evaluate the expectation <e>.[3] This augmentation of the top state causes the Soar I/O interface to generate the query, and a canned answer to the query (corresponding to the experimenter's answers) is retrieved by external Lisp code and deposited as an evaluation of <e> in Soar working memory. Production P6 then detects that this evaluation has been generated, and consequently the check-expectation-failure operator can be terminated in the design-instruction problem space:

[3]The production actually used in the simulation includes conditions that detect the specific context in which the ask operator is being applied. We do this so that we can include canned text in the production to mimick the actual statements (or paraphrases) of Subject W's questions. This is used purely as an aid for tracing and debugging the simulation.

P6: design-instruction*check-expectation-failure*expectation-acceptable*reject
IF in the goal context <g>
the problem space <p> is design-instruction
the operator is check-expectation-failure <q> for expectation <e>
and there is an acceptable evaluation of <e>
THEN the preference for <q> is reject in context <g>, problem space <p>, and state <s>

Note that production P6 applies only to information that is available in the goal context in which the original impasse on the check-expectation-failure operator occurred. This production does not detect anything about how the evaluation occurred, only that the operator has been achieved. In the following decision cycle, Soar terminates the check-expectation failure operator. Because the operator has been achieved, the subgoal context generated by the no-change operator impasse is removed.

In the same decision phase, Soar finds that the previously proposed specify operator is still acceptable for the top-level goal context, and that operator is then selected. Once again, there is no knowledge immediately available in the system that will implement the specify operator directly to generate the desired instructional specification, and consequently a no-change impasse occurs. Production P7, which is very much like production P4, sets up the problem space and initial state to resolve the impasse:

P7: specify*space&state&desired
IF in the goal context <g> the problem space is undecided
the context occurred because of a no-change impasse on operator <q>
the supergoal context <sg> has desired state <d>
<q> is a specify operator
THEN create a specify problem space <p>
the preference for <p> is acceptable in context <g>
create new state <s>
the preference for <s> is acceptable in context <g>
augment <g> with desired state <d>

Production P8 is then applied to augment the state with information about the specification that the specify operator is producing:

P8: specify*space*augment-result&type
IF in the goal context <g>
the problem space <p> is specify
the state is <s>
the specify operator <q> in the supergoal context is to specify result <r>
of type <t>
THEN augment state <s> with result <r> and type <t>

The no-change impasse that occurs on the specify operator, and the execution of P7 and P8 to create a problem space to resolve the specify impasse, occur with great frequency throughout the simulation.

Proposal of the General Design Plan

Following the application of productions P7 and P8, the simulation is in a specify problem space, with an initial state that is augmented with an identifier for the desired result and the type of specification desired (an instructional specification). At this point the

simulation decomposes the instructional specification into subspecifications that can be attacked separately. Although Subject W did not state this decomposition explicitly, his subsequent behavior and output conforms to the simulation's plan, and the plan is consistent with the steps of prescriptive instructional design methods (Mager, 1988). Other protocols in our database make explicit reference to similar plans. We assume that such general plans arise through the general training and enculturation of professional instructional designers (Goel & Pirolli, 1989).

More specifically, production P9 applies to propose a *decompose-specification* operator:

P9: specify*decompose-specification*instructional-specification*acceptable
IF in the goal context <g>
the problem space <p> is specify
the state is <s> containing a result <r> and type instructional-specification
THEN create a decompose-specification operator <q>
with a result <r> of type <t>
and subspecifications
of types expectations, target-population, task-analysis, and lesson-plan
with expectation <e>, target-population <tp>, task-analysis <ta>, and lesson-plan <lp>
the preference for <q> is acceptable in context <g>, problem space <p>, and state <s>

In general, productions similar to P9 propose such decompose-specification operators after matching the type of specification desired, and the operator is augmented to indicate that the specification may be decomposed into specific types of subspecifications. Each subspecification will be attacked by separate specify operators that often recursively invoke processing in separate specify problem spaces.

During the same elaboration phase in which P9 is executed, two other operators are proposed. An acceptable preference is also generated for a *get-information* operator. The purpose of this operator, if selected and applied, is to collect information that might be relevant to developing or decomposing a specification. However, the simulation also contains a production that recognizes when a decompose-specification and a get-information operator are simultaneously proposed, and assigns a reject preference to the get-information operator. In this way, decompositions get carried out if immediately possible, otherwise the default action is to collect more information. The second operator that is proposed is an *assemble-specification* operator. The purpose of this operator is to recognize when a set of subspecifications has been developed and the operator creates a specification object in working memory, which serves to mark the fact that a specification has been completed. The assemble-specification operator is assigned a *worst* preference so that all other acceptable operators are given higher priority. More generally, this example illustrates how the proposal of operators and their prioritization can be carried out by separate pools of knowledge in Soar operating in concert and largely in parallel during an elaboration phase.

After the Soar decision phase selects the decompose-specification operator and installs it in the current context, the application of the operator is carried out by the parallel execution of instantiations of a production that augments the state to indicate that the desired specification can be decomposed into a specific set of subspecifications. Parallel execution of another production also copies this information about the decomposition onto the the specify operator that caused the current impasse. Another production registers when the subspecifications proposed by the decompose-specification operator are instantiated on the

current state, and the operator is assigned a reject preference when all subspecifications have been copied onto the state.

As augmentations pointing to the desired subspecifications appear on the state, another production creates and proposes acceptable preferences for a specify operator for each subspecification. Sequencing of the operators is achieved by productions that propose *better* preferences, which is a binary transitive relation indicating that one operator should be selected above another. The sequence in which the decomposition of the instructional specification is processed in our simulation is depicted in Figure 7.

Insert Figure 7 about here

In summary, the specification of the instructional design is carried out by a particular specify-operator. That specify-operator is implemented by processing in a subproblem space. The processing in that subproblem space is initiated by a decompose-specification operator. Parallel processes, implemented by productions, implement the decomposition by simultaneously elaborating the state associated with the subproblem space and the specify-operator that caused the subproblem space. The decompositions that appear on the state associated with the subproblem space immediately trigger the proposal of new specify-operators. An assemble-specification operator waits until all subspecifications are completed and then is triggered to indicate that the specification task is done.

Specification of Customer Expectations and the Target Population

The impasse caused by the selection of a specify operator for customer expectations (Figure 7) again causes the selection of a specify problem space to implement the operator and overcome the impasse. A decompose-specification operator is selected and applied such that customer expectations are broken down into subspecifications for delivery constraints and instructional objectives (see Figure 8). Specify operators proposed for these subspecifications result in impasses and each is implemented in separate specify problem spaces.

Insert Figure 8 about here

The specification of delivery constraints is achieved by *specify-desired-constraint* operators. These operators are proposed by productions that recognize that the current specification involves information about constraints contained in the desired state specification. The application of the operators involves writing out the desired constraints as parts of the desired specification (and writing them as output to the Soar I/O interface). An assemble-specification operator recognizes when the listing of delivery constraints is complete by generating an internal *specification object* that contains the specification information on the current state. The application of the specify operator for the delivery constraints is terminated by a production that proposes a reject preference for the operator when the relevant specification object is generated, the impasse is terminated, and the next specify operator is selected for application.

At this point in the protocol, Subject W asked for an evaluation of several possible objectives for the course, and the experimenter offered a list of proposals:

S: Within this training, the customer is going...

What is the end expectation?

Are they going to be expected to be able to use graphics?

Are they going to be able to use ah, any of the add features, ah, the sums, ah

E: Here, let me give you a brief description of the kinds of tasks they'll want them to be able to do.

S: Right

E: They'll want them to be able to create and edit memos and documents.

S: Ok [writing down objective]

E: They'll want them to be able to do mail.

S: Ok [writing down objective]

E: They're going to want them to be able to tables and graphics, and that kind of stuff...

S: Ok [writing down objective]

E: The other thing is they are probably going to have a feel for the whole, I guess filing system, or just they're way around where things are, on the machine...

S: Ok [writing down objective]

Apparently, Subject W was gathering additional information to refine the objectives of the word processing course and used his own background knowledge to propose several possibilities for evaluation and clarification. The experimenter then proposed a more specific set of objectives.

The Soar simulation hits an impasse on the specification of instructional objectives, because there is no chunked background knowledge about this specific task. There is also no chunked knowledge available to decompose this particular specification in the specify problem problem space that is selected for this impasse. Thus, the default get-information operator is selected, which causes an additional impasse because there is no chunked knowledge to retrieve about the objectives for this design. We assume that all of these impasses are just a result of the fact that Subject W has never experienced this particular design problem.

Productions select a *get-information* problem space to implement the get-information operator. Three kinds of operators are proposed and applied in this problem space. First, an *elaborate-information* operator is proposed. Such operators are viewed as cues to retrieve declarative information from long-term memory to elaborate some item. Cue-specific productions apply in parallel during an elaboration phase to augment the item

with additional information. Next, *evaluation* operators are proposed and selected to evaluate whether the new information is relevant to the current design. Finally, *ask* operators may be proposed and selected to get information from external sources. As information is evaluated as relevant, it is propagated back to the the specification problem space. The result of the get-information operator is an elaboration of the the objectives list to include the creation and editing of documents, mailing, the creation of tables, graphics handling, file manipulation, and spell checking. An assemble-specification operator creates a specification object containing this information to indicate that the objectives have been completed.

Having completed the specification of customer expectations, the simulation turns to the specification of the target population. Here the simulation again uses specify-delivery-constraint operators to copy information about the secretaries from the desired state information.

Specification of the Task Analysis

After specifying features of the target population, Subject W turned to a specification of the task analysis for creating and editing documents. Using the Viewpoint manual, he developed a list of the the various system concepts and procedures that the secretaries would have to learn. The main process was one of looking at headings in the table of contents for the manual, and then transferring the heading to the task listing, if Subject W thought the topic should be part of the the training. Roughly then, the manual served to cue the recall of information from long-term memory (Subject W rarely looked at the contents of the manual), and evaluations of the relevancy and importance of the topics to the objectives of teaching the secretaries about document creation and editing.

Our Soar simulation of Subject W's task analysis is based on the hypothesis that Subject W was constructing a mental representation of a generic plan that secretaries would follow in creating and editing documents. After hitting an impasse on the specification of a task analysis, the simulation moves into a get-information problem space. Another impasse is hit when an elaborate-information operator fails to retrieve any information. A *comprehend-input* problem space is proposed, based on the recognition that there is an external source (the Viewpoint manual) containing relevant information.

Our simulation scans through the table of contents for the Viewpoint manual and makes use of the indenting as an indication of subtask and subconcept relations. Each item in the index is input and serves as a cue for additional elaboration by productions that augment the item with long-term declarative knowledge. Most importantly, system concepts are elaborated with the roles they can play in various tasks (e.g., the role a keyboard can play). The system then proposes these tasks and concepts as part of a mental model for the secretaries' tasks. The tasks also come with prerequisite links (i.e., precondition and enabling links) and productions use this information to create a coordinated structure of tasks linked to system concepts that are relevant to the performance of the tasks. The plan that is developed is depicted in Figure 9.

Insert Figure 9 about here

More specifically, the goal of the task analysis is to create a plan-like outline of the procedure involved with creating and editing ViewPoint documents. Items are read in from the manual's table of contents, are labeled as a task or concept, and productions attempt to merge the items into the developing plan outline. If a content item is a task, productions

evaluate whether or not it is a prerequisite of task that is already in the plan, or if it is directly associated to the performance of the task. If so, the item is merged with the developing plan.

If a content item is a concept, the simulation determines if the concept may play a role in any tasks. The simulation then determines if any such task is already part of the developing plan and, if so, the content item is accepted and merged with the plan. If a related task is not in the current plan, the simulation tries to determine whether or not this new task is related to or a prerequisite of a task already in the outline. If so, then both the concept item and the new task are accepted. If the concept and task are unrelated to the evolving plan, then they are rejected.

To an approximation, this part of our simulation bears resemblance to work on the integration of instruction comprehension and task performance in Soar (Lewis, Newell, &Polk, 1989). The purpose of that work is to model data from psychological experiments. The models process natural language inputs which describe the task to be performed. The output of these comprehension processes is a *behavior model*. The behavior model is then interpreted to propose problem spaces and operators to perform the desired task. In our simulation, comprehension processes operate on the Viewpoint manual table of contents to produce a plan-like behavior model of the tasks secretaries will perform, and this behavior model is used to generate a task specification.

Specification of a Lesson Plan and Course Overview

Subject W's task was to design the first session of instruction in detail. After developing the task analysis, Subject W began work on the lesson plan for the first session. As a subtask in this lesson plan, Subject W developed a general outline of the six lessons.

In our simulation, following the specification of the task analysis, a decompose-specification operator is selected to carry out the specification of the first session (see Figure 10). The operator decomposes the lesson plan for the first session into specifications of (a) an introduction, (b) the relevance of the course to the students, (c) the objective of the session, and (d) a course overview. The general structure of this plan is highly similar to those proposed in the instructional design literature (Mager, 1988).

Insert Figure 10 about here

Subject W in specifying the introduction simply writes "Session # 1 - 1 Hour." Our simulation also has a highly specific production that mimicks this specification. The specification of the course relevance is handled in its own specification problem space, and it is decomposed into three subspecifications: (a) specification of features of working with ViewPoint that will be new to the secretaries, (b) specification of important features that are in ViewPoint that the secretaries will be familiar with, and (c) and example of ViewPoint use. We hypothesize that this decomposition is standard for Subject W and probably arose from previous experience in motivating the introduction of new technologies into the workplace. Get-information operators retrieve the relevant information from memory. The specifications include (a) every secretary will have their own terminal and these will be networked, (b) ViewPoint supports on-line text-editing and will come with a printer, and (c) the instructional manual itself is a product of ViewPoint. The session objective is

specified by Subject W simply as "explain expectations after training," and similarly the simulation has a highly specific operator that outputs the same specification.

In order to specify the course overview, Subject W had to first develop a more specific plan for the entire six sessions. Subject W reviewed the previously specified task analysis and objectives list and arranged the material into the various sessions. In our simulation, the task analysis is reread through Soar I/O, and each item is evaluated to determine if it will fit into the first session. Then the system generates a curriculum structure containing slots for each of the six sessions, with the first session labelled as a familiarization session, and the last session labelled as a review session, which corresponds to the general plan for the sessions developed by Subject W. The system then attempts to assign course objectives, and components of the task analysis not included in Session 1, to Sessions 2-6.

Subject W rearranged the course outline several times. Our simulation captures these rearrangements using a relatively simple strategy that seems to correspond to that of Subject W: (a) the system simply assigns the next input objective or task analysis item into the next free session slot, and (b) if the next input item happens to be a prerequisite for an already assigned session objective, the input item is inserted before the objective for which it is a prerequisite.

Comparison to Other Protocols

While Subject W's protocol has been the major focus of the investigation, all nine subject's protocols were analyzed for similarities to Subject W's protocol. Also, two other protocols were coded in detail using in the scheme discussed earlier.

One of the obvious similarities between all the subjects was their use of course and lesson outlines in the design process. All subjects started their designs with some basic list of objectives or tasks, transformed and refined these lists into an outlines of the sessions, and expanded simple outlines into more complex ones.

Another conspicuous similarity was the way almost all of the subjects split the instructional content into the six sessions (one subject did not define the sessions). The standard breakdown of the content was like Subject W's:

- Session 1: Intro
- Session 2: Basic document editing
- Session 3: Filing, Mail, and/or Printing
- Session 4: Graphics or Tables
- Session 5: More Graphics or Tables
- Session 6: Review/Practice or advanced topics

Five of the subjects were in the experimental group requested to do their design a in a depth-first manner (like Subject W). While the details of the first lesson differed to some degree between the subjects, there was similarity in the format of the lesson. The subjects generally followed the prescriptive style of a introductory lesson set forth by Mager (1988); they each developed sections corresponding to (a) an introduction, (b) course relevance, (c) lesson objective, and (d) a course overview. All of the subjects explicitly stated the need and importance of defining objectives before designing instruction.

General Discussion

A striking general characteristic of our Soar simulation is the substantial amount of activity that was devoted to problem structuring. That is, much of the activity involved the refinement or specification of the desired state, task constraints, and in many cases there were no immediately obvious operators for producing a specification. In such cases, the problem space was relatively ill-structured and the activity was focused on making it more well-structured. Much of the structure for problem solving was developed by accessing and evaluating information from long-term memory and external sources. This is consistent with protocol analyses (Goel & Pirolli, 1991) of instructional design, architectural design, and mechanical design, that indicated that approximately 25% of the statements were devoted to problem structuring, whereas protocols culled from the literature on well-structured problem solving showed an average of less than 1% of the statements devoted to problem structuring.

The critical role of information (or knowledge) access, interpretation, and manipulation in ill-structured tasks was recognized by Simon (1973) and, as noted in our introduction, he suggested that classical problem solving analyses could be extended to incorporate such processing. Indeed, our Soar simulation performed such problem structuring using problem space search. It is the nature of the Soar architecture that cases of insufficient structure in a problem space--or a lack of *task implementation knowledge*--are treated as impasses that can be addressed as new problems to be solved.

Although problem structuring was addressed to some extent by Simon and Hayes' (Hayes & Simon, 1974; Hayes & Simon, 1976) UNDERSTAND research, their effort focused on the comprehension of relatively simple instructions for well-structured puzzle problems that were semantically impoverished. In general, however, the access, interpretation, and manipulation of information involved in structuring ill-structured, semantically rich, tasks has not received much attention, and it clearly constitutes a large residual category of unaddressed phenomena in problem solving models. Indeed, many, if not most, of the difficult problems faced every day in tasks such as buying a house, figuring out what to do in graduate school, finding a mate, or managing a business, substantially involve the collection and interpretation of information in order to understand and choose the relevant problem spaces and associated actions in those spaces.

In our Soar simulation, the structuring of the instructional design specification was substantially determined by the identification and refinement of the objectives, conditions, and constraints of the instruction, and the development of a mental model of the tasks to be performed by secretaries. The identification and refinement of instructional objectives, conditions, and constraints of the instruction were partly carried out through interaction with the experimenter/client. Such interactions are probably a (somewhat pale) reflection of the sorts of multiagent interactions that usually characterize design activities. More refined interpretations of the statement of the design objectives, conditions, and constraints were offered up for external evaluation or further clarification. Indeed, this interactive process of refining and aligning the interpretation of meanings from the design brief might be viewed as a simple form of *negotiation* about *commitments* between two agents in the parlance of distributed artificial intelligence (Gasser, 1991; Hewitt, 1991).

External sources of knowledge used to structure problem solving were used both as knowledge generators and tests in the construction and selection of design elements. For instance, the text editor manual was used to cue recall of concepts and tasks relevant to the use of the text editor. These were tested and integrated into a plan-like model of tasks

appropriate for novice secretaries. In this case, external sources were essentially acting as knowledge generators. A contrasting example occurs earlier in the simulation when a list of possible secretary tasks was generated from long-term memory and these items are proposed to the experimenter/client for verification. In this case, internal long-term memory was acting as a knowledge generator and external sources were used as search control tests.

Examination of the protocol data summarized in Figures 2, 3, and 4 suggests that the design process can be viewed as a tension among an ordered attack on specific areas of content, some degree of opportunism, and attention to interrelations among content modules. This observation has been noted in other empirical studies of design (Goel & Pirolli, 1991; Guindon, 1990; Kant, 1985; Ullman, Dietterich, &Stauffer, 1988). It has been argued (Goel & Pirolli, 1989; Goel & Pirolli, 1991) that this characteristic tension in design emerges because design problems are basically large-scale constraint-satisfaction tasks that have to be handled by a human processor with limited information-processing capacities.

One problem-solving method that addresses the tension between managing complexity and attending to details is the constraint-posting hierarchical planning developed in MOLGEN (Stefik, 1981). MOLGEN represents plans as abstract operators and states. These plans are refined through decomposition. Constraints express relationships among plan variables, constrain the selection of entities to incorporate into plans, express commitments, and indicate interactions among subproblems. Constraints are formulated within separate parts of the problem during hierarchical planning and then propagated among the subparts to bring together requirements and to coordinate nearly independent subproblems.

At some level of approximation, our Soar simulation implements a variant of MOLGEN's mix of hierarchical planning and constraint posting. The instructional design problem itself is proposed as a constraint satisfaction problem. The control strategy presented in Figure 5 involves problem decomposition operators which carry out a form of hierarchical planning. The context-sensitive elaboration of these operators, sometimes cued by information acquired through the get-information operators, is a form of constraint formulation and propagation.

On closer inspection, our Soar model can be viewed as implementing several methods that achieve this mix of hierarchical planning and constraint satisfaction. Our Soar model can be viewed as sometimes using problem decomposition via precompiled plans. For instance, the simulation contains a chunk that decomposes the task of producing an instructional design into subtasks involving the specification of customer expectations, target population, task analysis, and lesson plan. In other cases, the simulation can be viewed as decomposing and solving problems by generate and test. For instance, in developing a task analysis, the external environment and long-term memory are used to generate possible subtasks and concepts relevant to creating and editing documents. These concepts and subtasks have certain constraint relations among them that must be tested, in addition to testing their value for novice secretaries. To an approximation, the Soar model might be viewed as employing a kind of transformational approach to design, in which one kind of specification representation is transformed into another. The transformation of the task analysis into lesson specifications is one such example. Finally, the use of background knowledge to structure design might be viewed as a case-based approach. The checking of expectation failures due to background knowledge about clients buying office systems is an example of such case-based reasoning.

The notion that design can be addressed by diverse methods is recognized by many in the AI community (Brown & Chandrasekaran, 1989; Chandrasekaran, 1990; Maher, 1990), and Chandrasekaran (Brown & Chandrasekaran, 1989; Chandrasekaran, 1990) has argued for some time that AI design systems should be constructed to use heterogeneous methods rather than any single approach. Brown and Chandrasekaran's (1989) *generic task* framework, in which they develop such a heterogeneous model of routine design, is in fact quite similar to the multiple problem space framework underlying Soar.

The protocol analysis and Soar simulation suggest that even simple cases of nonroutine design involve substantial collection, evaluation, and integration of information. The analyses and simulation also suggests that such design involves a heterogeneous collection of methods achieving a tension between hierarchical problem decomposition and global constraint satisfaction. Problem solving approaches to design that involve a uniform method, or closed-world semantics would seem to face intractable complexities in tackling such problems. Interestingly, the extension of the problem space search model to a multiple problem-space architecture with diverse methods and the ability to incorporate new task-implementation knowledge seems to be sufficient to deal with such design problems.

References

Anderson, J. R. (1983). *The architecture of cognition* . Cambridge, MA: Harvard University Press.

Brown, D. C., & Chandrasekaran, B. (1989). *Design problem solving* . San Mateo, CA: Morgan Kaufmann.

Chandrasekaran, B. (1987). *Towards a functional architecture for intelligence based on generic information processing tasks*. Morgan Kaufman,

Chandrasekaran, B. (1990). Design problem solving: A task analysis. *AI Magazine, 11*, 59-71.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal reports as data* . Cambridge, MA: MIT Press.

Gasser, L. (1991). Social conceptions of knowledge and action: DAI foundations and open systems semantics. *Artificial Intelligence, 47*, 107-138.

Goel, V., & Pirolli, P. (1989). Motivating the notion of generic design within information-processing theory: The design problem space. *AI Magazine, 10*, 19-36.

Goel, V., & Pirolli, P. (1991). *The structure of design problem spaces* (DPS-3). University of California.

Greeno, J. G., & Simon, H. A. (1988). Problem solving and reasoning. In R. C. Atkinson, R. J. Hernstein, G. Lindzey, & R. D. Luce (Ed.), *Steven's Handbook of experimental psychology* (pp. 589-672). New York: Wiley.

Guindon, R. (1990). Designing the design process: Exploiting opportunistic thoughts. *Human-Computer Interaction, 5*, 305-344.

Hayes, J. R., & Simon, H. A. (1974). Understanding written problem instructions. In L. W. Gregg (Ed.), *Knowledge and cognition* (pp. 167-200). Hillsdale, NJ: Lawrence Erlbaum Associates.

Hayes, J. R., & Simon, H. A. (1976). Understanding complex task instructions. In D. Klahr (Ed.), *Cognition and Instruction* (pp. 269-286). Hillsdale, NJ: Lawrence Erlbaum Associates.

Hewitt, C. (1991). Open Information Systems Semantics for Distributed Artificial Intelligence. *Artificial Intelligence, 47*, 79-106.

Kant, E. (1985). *Understanding and automating algorithm design*. Los Altos, California: Morgan Kaufman, 1243-1253.

Lewis, R. L., Newell, A., & Polk, T. A. (1989). Toward a Soar theory of taking instructions for immediate reasoning tasks. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, (pp. 514-521). Hillsdale, NJ: Lawrence Erlbaum Associates

Mager, R. F. (1988). *Making Instruction Work* . Belmont, CA: Lake.

Maher, M. L. (1990). Process models for design synthesis. *AI Magazine, 11*, 49-58.

Mostow, J. (1985). Toward better models of the design process. *AI Magazine, 6*, 44-57.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence, 40*, 119-184.

Newell, A. (1990). *Unified theories of cognition* . Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving* . Englewood Cliffs, NJ: Prentice Hall.

Reigeluth, C. M. (1983). Instructional design: what is it and why is it? In C. M. Reigeluth (Ed.), *Instructional-design theories and models* (pp. 3-36). Hillsdale, NJ: Lawrence Erlbaum.

Reitman, W. R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M. W. Shelly, & G. L. Bryan (Ed.), *Human judgements and optimality*. New York: Wiley & Sons.

Reitman, W. R. (1965). *Cognition and thought: An information-processing approach* . New York: Wiley.

Simon, H. A. (1973). The structure of ill-structured problems. *Artificial Intelligence, 4*, 181-204.

Stefik, M. (1981). Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence, 16*, 111-140.

Ullman, D. G., Dietterich, T. G., & Stauffer, L. A. (1988). *A model of the mechanical design process based on empirical data* (Tech. Rep. DPRG-88-1).

Appendix

Information Types

- *Knowledge source*. Indication that someone or something will be a resource in making decisions for some purpose.
- *Student*. Statements about properties of individual or groups of students, or about members of the student population.
- *Constraint*. A constraint is an information type which fixes some feature about the general form of instruction. Constraints have the form <form-feature> [=|relation] [value|feature]. Figures 2, 3, and 4 pool together constraint statements and *resource* statements, which are about resources that can be used with the instruction.
- Design. This includes three kinds of statements:
    - *Design operator*. This information type refers to a statement about a design operation. Operators include things like specify, prioritize, sequence, decompose-task, locate, review, and gather-information.
    - *Design methodology*. A statement about design methods typically used, or used in certain circumstances.
    - *Design resource*. In these statements, the subject is discussing some resource that is used during the design. process.
- *Content*. A statement about the content or subject-matter that may be part of the instruction.
- *Session*. A statement about the content of a particular session.
- *Transaction*. A transaction is an instructional interaction (e.g., explain, tell, question), usually about some content and intended to meet some objective. The transaction information type is like the "content" type. A transaction is added when the subject presents a content element as an instructional transaction and writes it down in a course or session outline.
- *Sequence*. These are statements that relate to the ordering of the instruction.
- *Prerequisite*. A statement indicating some content is prerequisite for another.
- *Objective*. These statements indicate an instructional objective.
- *Principle*. A statement of an instructional principle.

- *Conditions*. These statements concern the state of the instructional situation at some point in time or over some time course (i.e. start, during, or after).

Knowledge Sources

- *Experimenter*. The experimenter is the source of the information.
- *Manual*. The manual is the source of the information.
- *Briefs*. The design brief or the task instructions is the source of the information.
- *Self*. The subject is the source of the information.
- *Derived*. The source of the information is Self, but s/he had to derive it from other knowledge.

Author Notes

This research has been supported by the Office of Naval Research, Cognitive and Neural Sciences Division, Contract No. N00014-88-K0233 to the Peter Pirolli. We would like to thank Mimi Recker for comments on an earlier draft of this report.

Figure Captions

Figure 1. An overview of the prescriptive design process model of Mager (1988).

Figure 2. A classification of the information types attended to by Subject W over the course of his protocol. See the text for details

Figure 3. The distribution of external knowledge sources over the course of Subject W's protocol.

Figure 4. The distribution of internal knowledge sources over the course of Subject W's protocol.

Figure 5. The general control structure of the specification problem spaces.

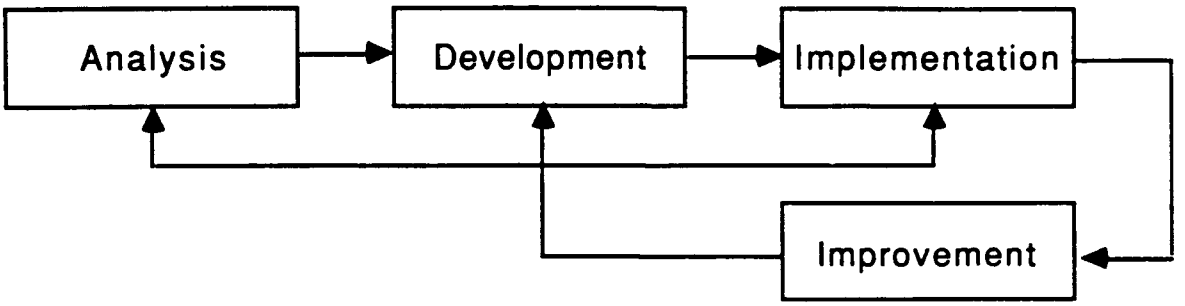
Figure 6. Checking an expectation failure before specifying the instructional design.

Figure 7. The decomposition of the the instructional design into a general plan.

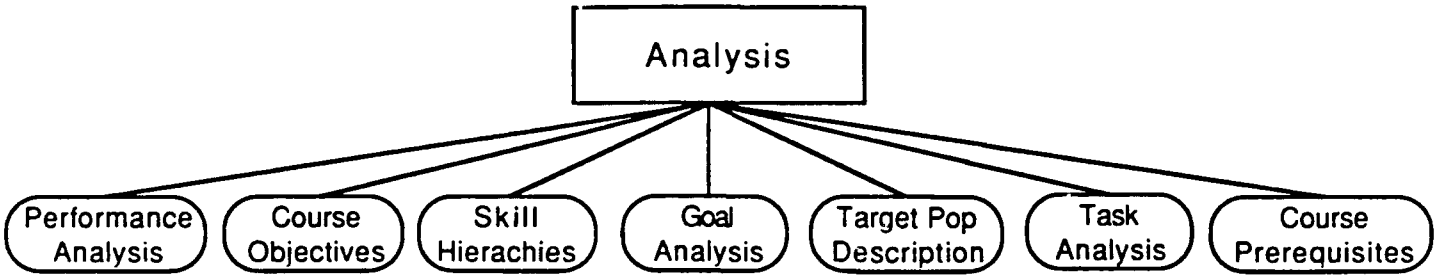
Figure 8. The decomposition of customer expectations.

Figure 9. The plan-like representation for the task analysis of task editing for novice secretaries.

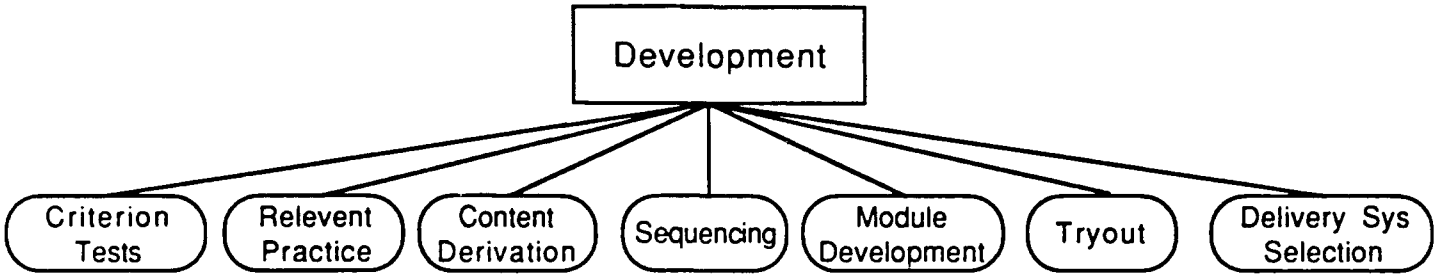
Figure 10. The decomposition of a lesson plan for an introductory lesson.



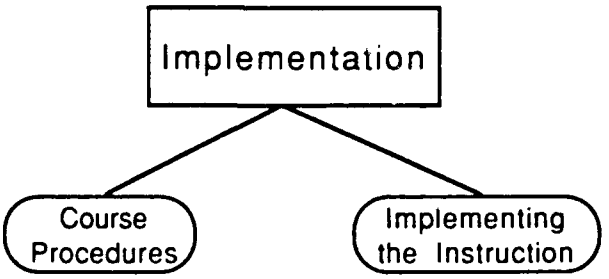
The Four Phases of The Instructional Process



Components of The Analysis Phase



Components of The Development Phase



Components of The Implementation Phase

Figure 1.

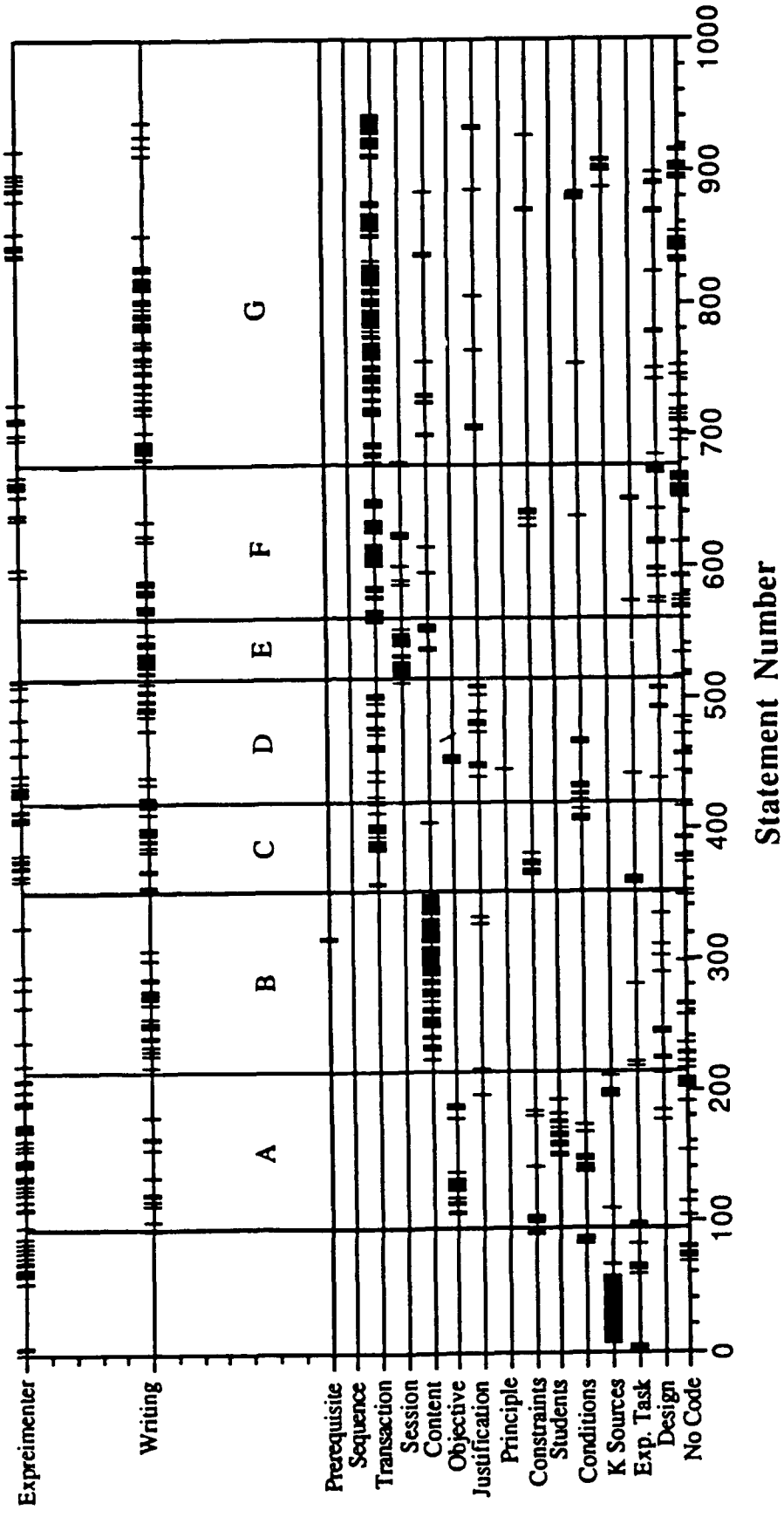

Expreimenter
Writing
A
B
C
D
E
F
G
Prerequisite
Sequence
Transaction
Session
Content
Objective
Justification
Principle
Constraints
Students
Conditions
K Sources
Exp. Task
Design
No Code
0
100
200
300
400
500
600
700
800
900
1000
Statement Number

Figure 2.

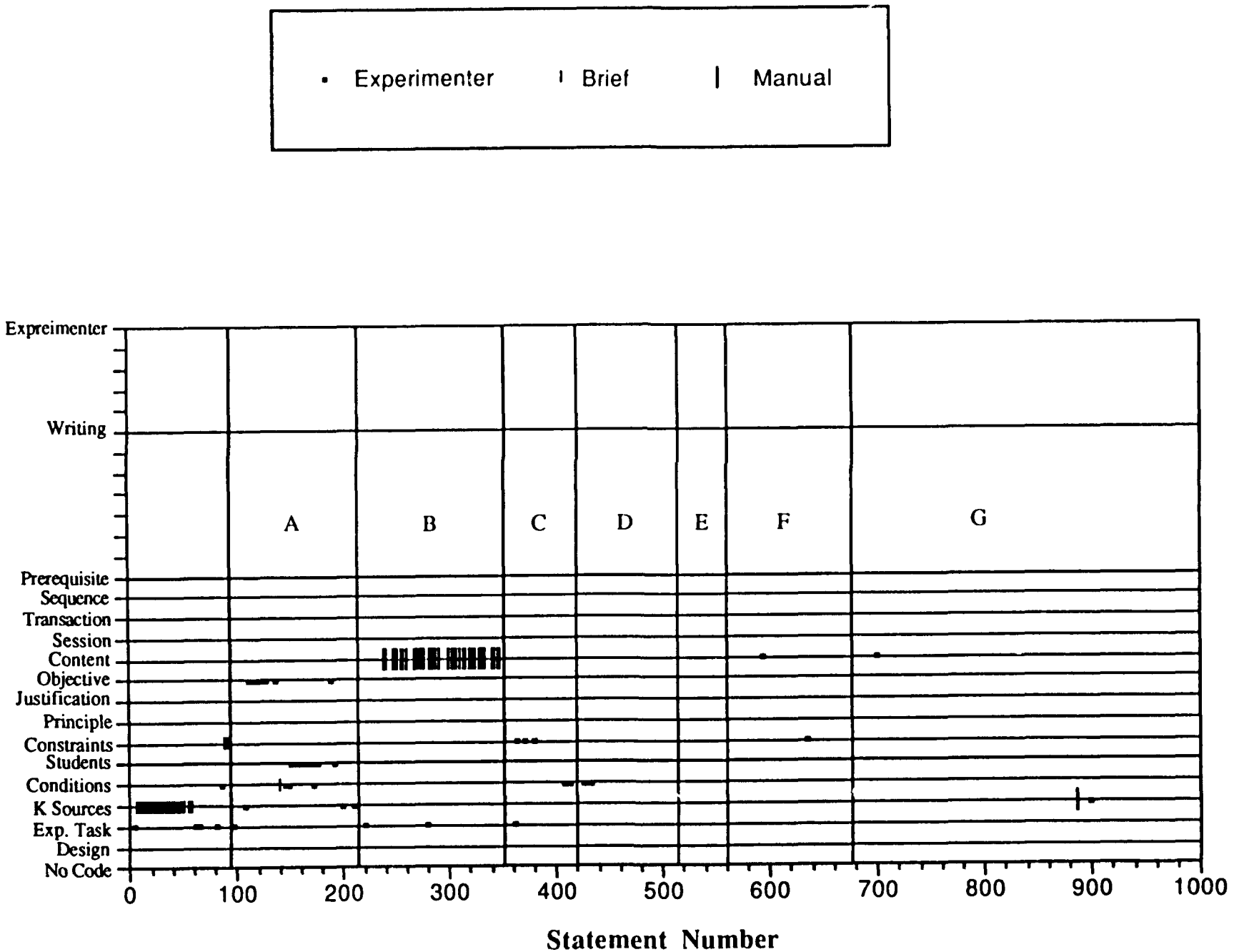
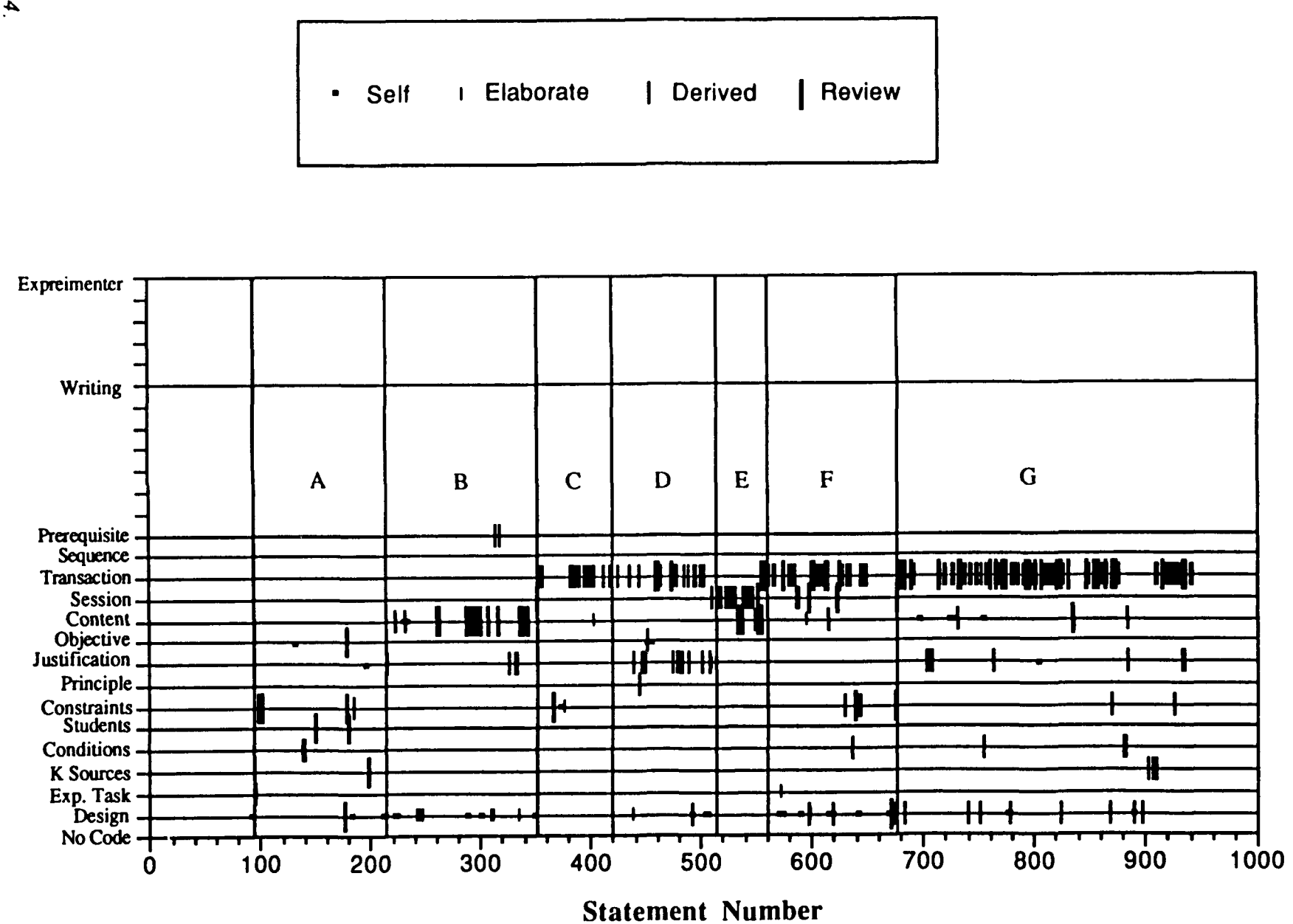

• Experimenter
Brief
Manual
Expreimenter
Writing
A
B
C
D
E
F
G
Prerequisite
Sequence
Transaction
Session
Content
Objective
Justification
Principle
Constraints
Students
Conditions
K Sources
Exp. Task
Design
No Code
0
100
200
300
400
500
600
700
800
900
1000
Statement Number

Figure 3.

Figure 4.


Self
Elaborate
Derived
Review
Expreimenter
Writing
A
B
C
D
E
F
G
Prerequisite
Sequence
Transaction
Session
Content
Objective
Justification
Principle
Constraints
Students
Conditions
K Sources
Exp. Task
Design
No Code
0
100
200
300
400
500
600
700
800
900
1000
Statement Number

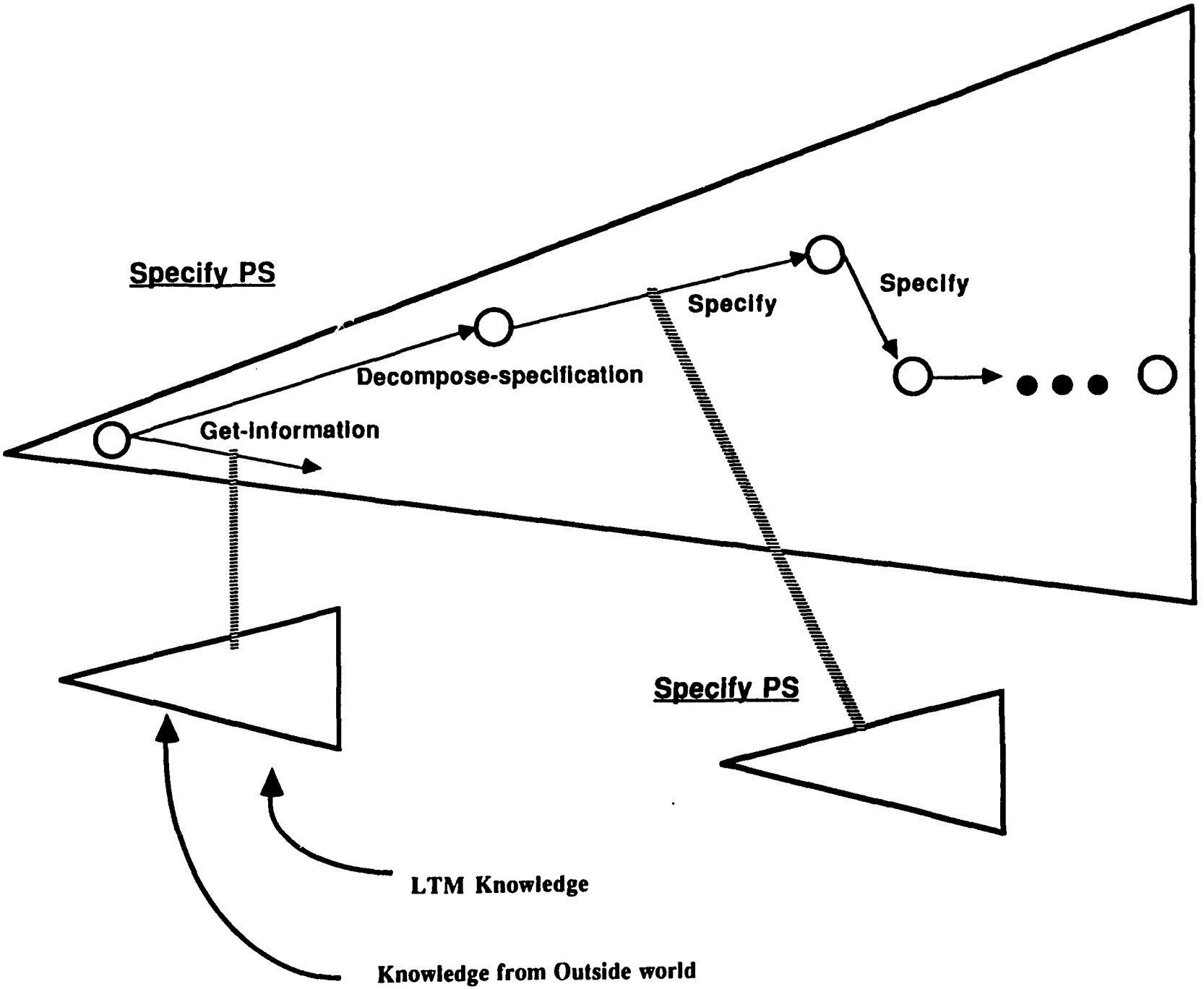

Specify PS
Specify
Specify
Decompose-specification
Get-information
Specify PS
LTM Knowledge
Knowledge from Outside world

Figure 5.

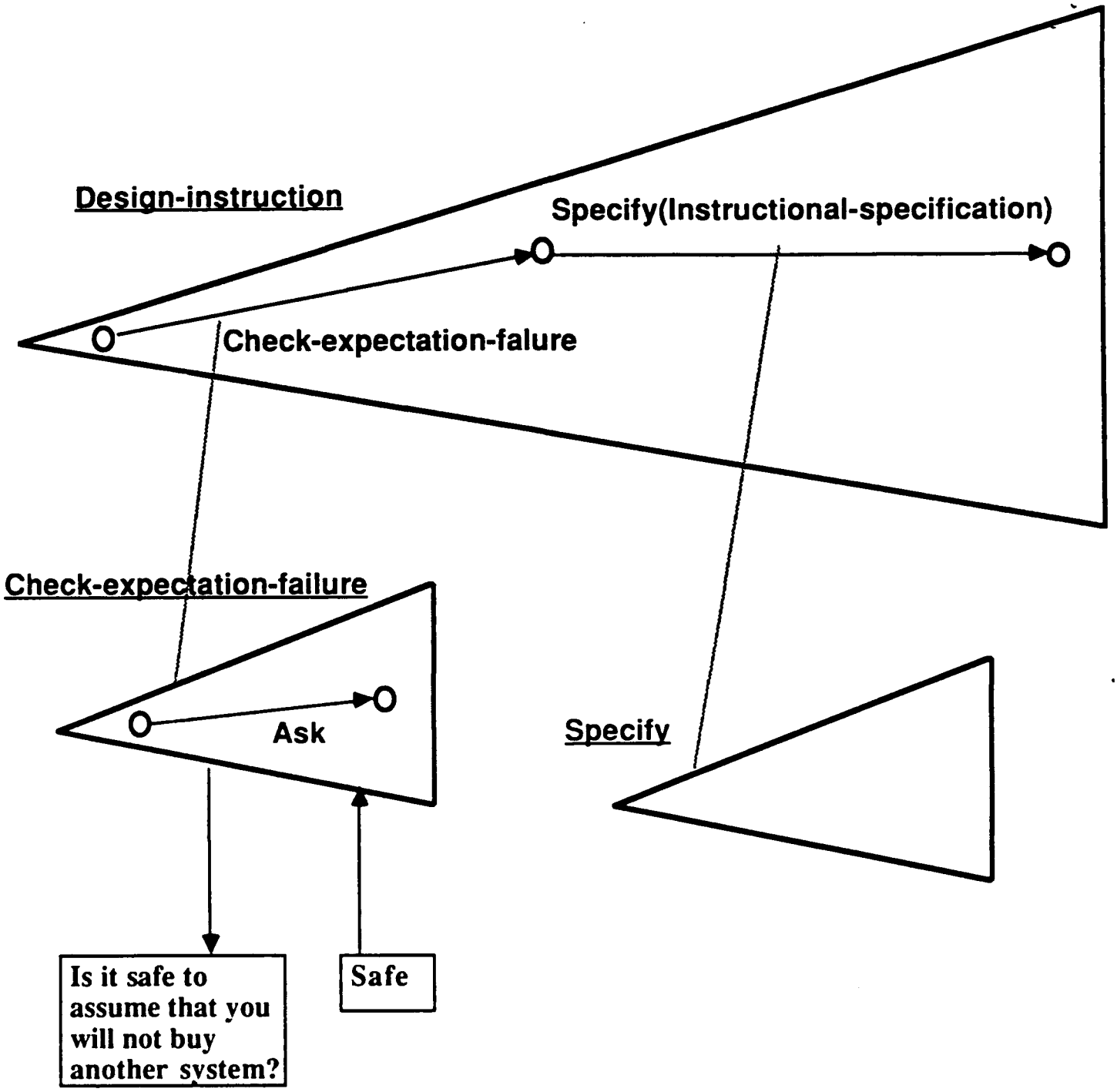

Design-instruction
Specify(Instructional-specification)
Check-expectation-falure
Check-expectation-failure
Ask
Specify
Is it safe to assume that you will not buy another system?
Safe

Figure 6.

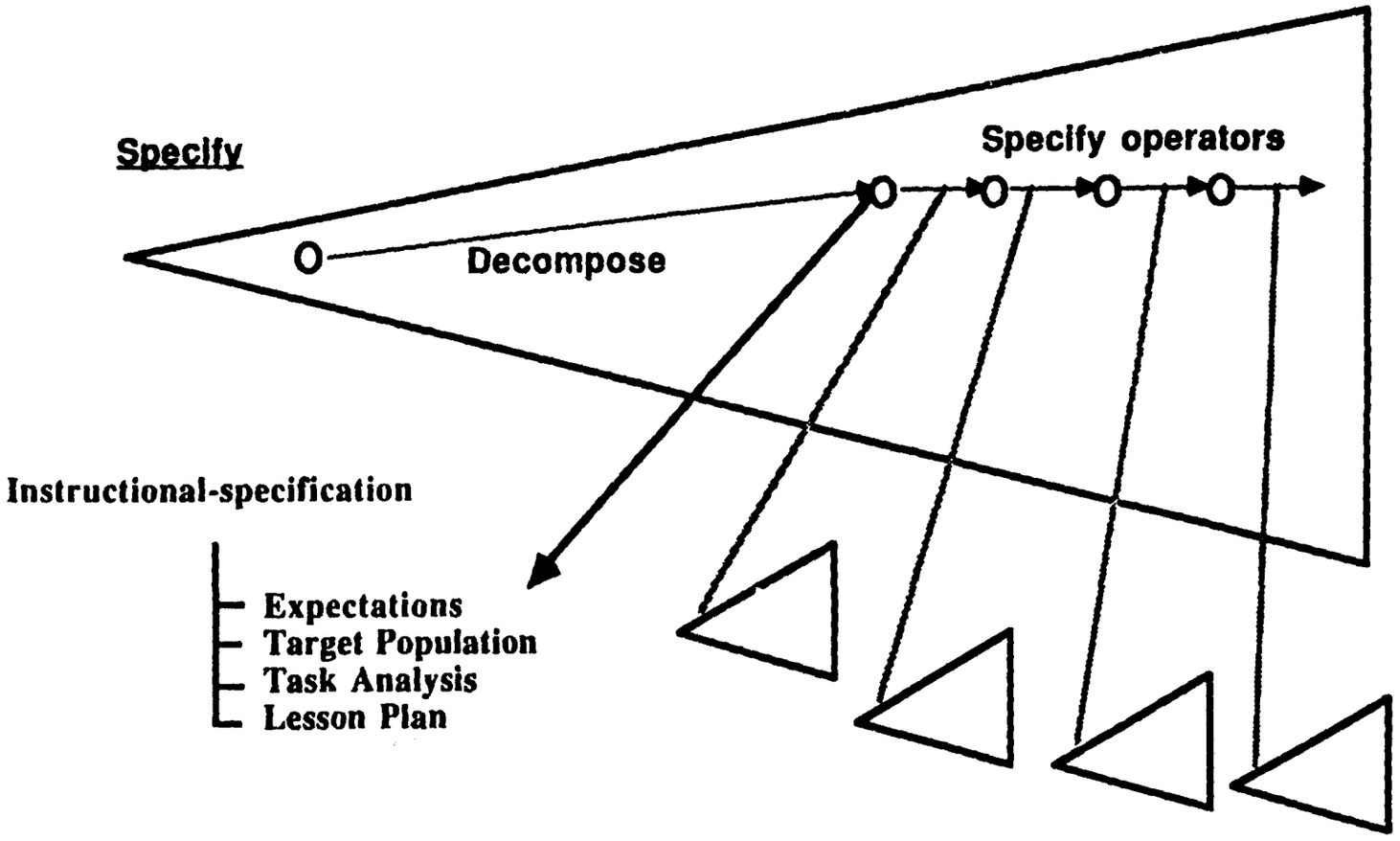
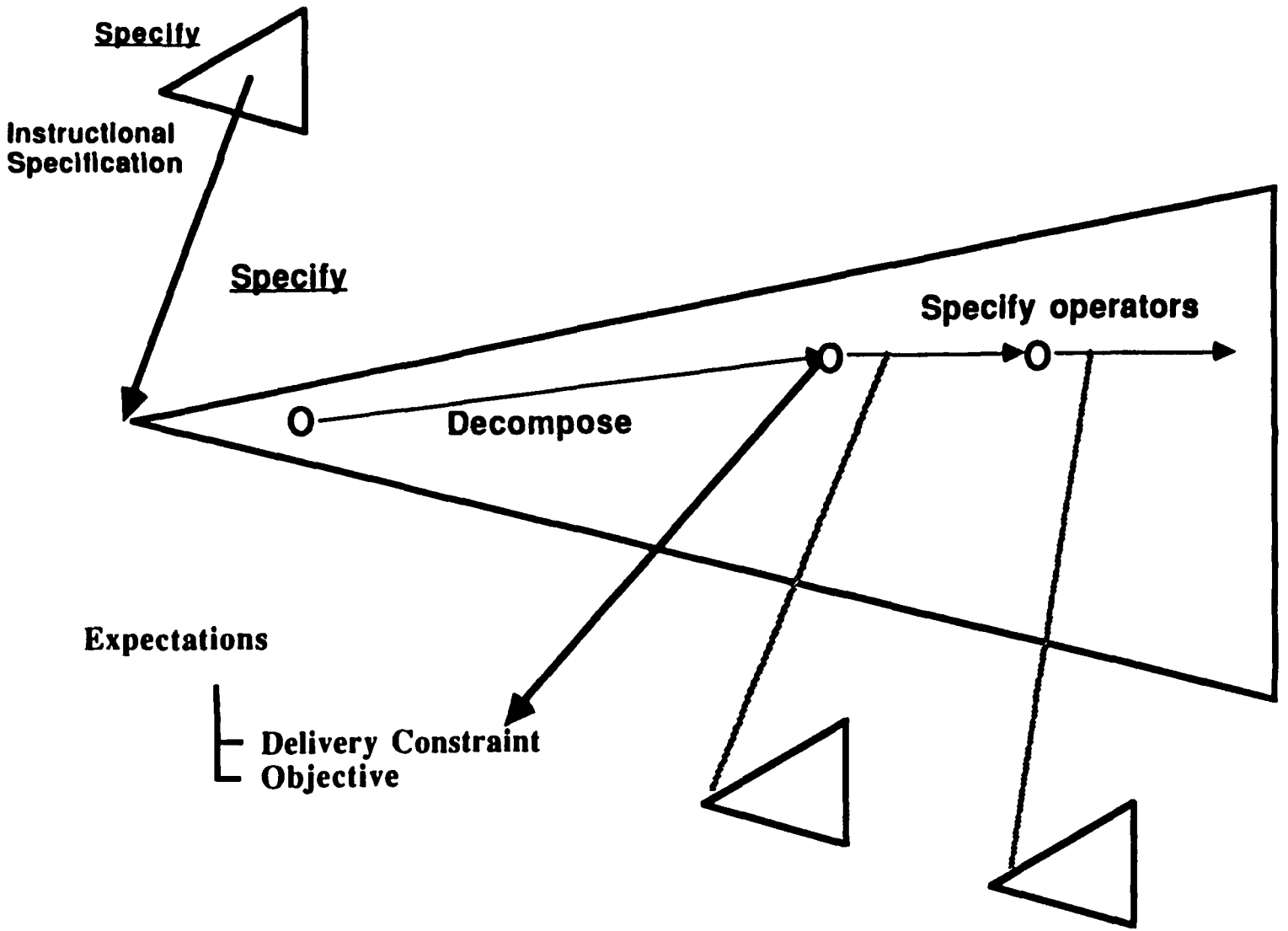

Specify
Specify operators
Decompose
Instructional-specification
Expectations
Target Population
Task Analysis
Lesson Plan

Figure 7.

Figure 8.


Specify
Instructional Specification
Specify
Specify operators
Decompose
Expectations
Delivery Constraint
Objective

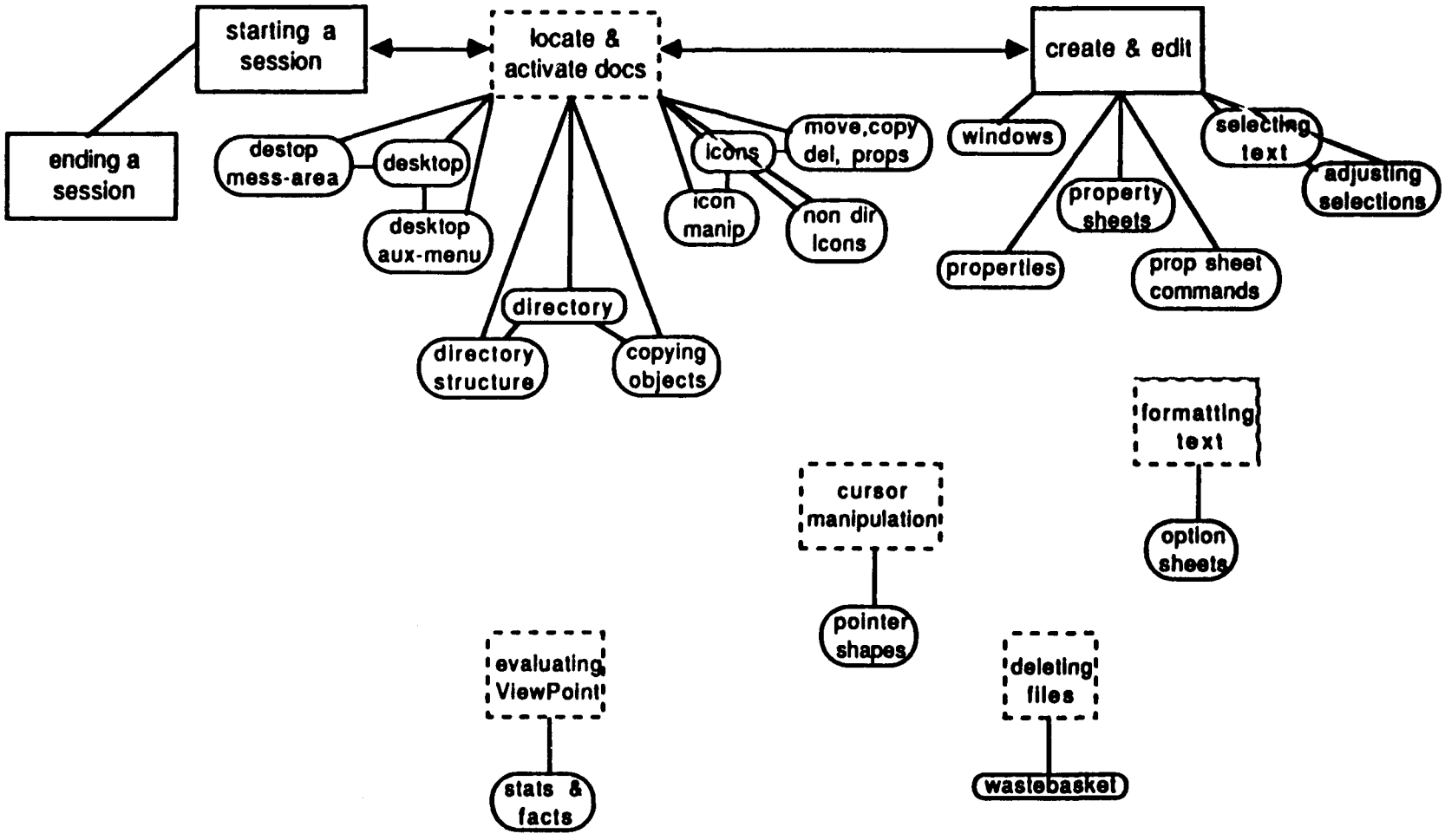
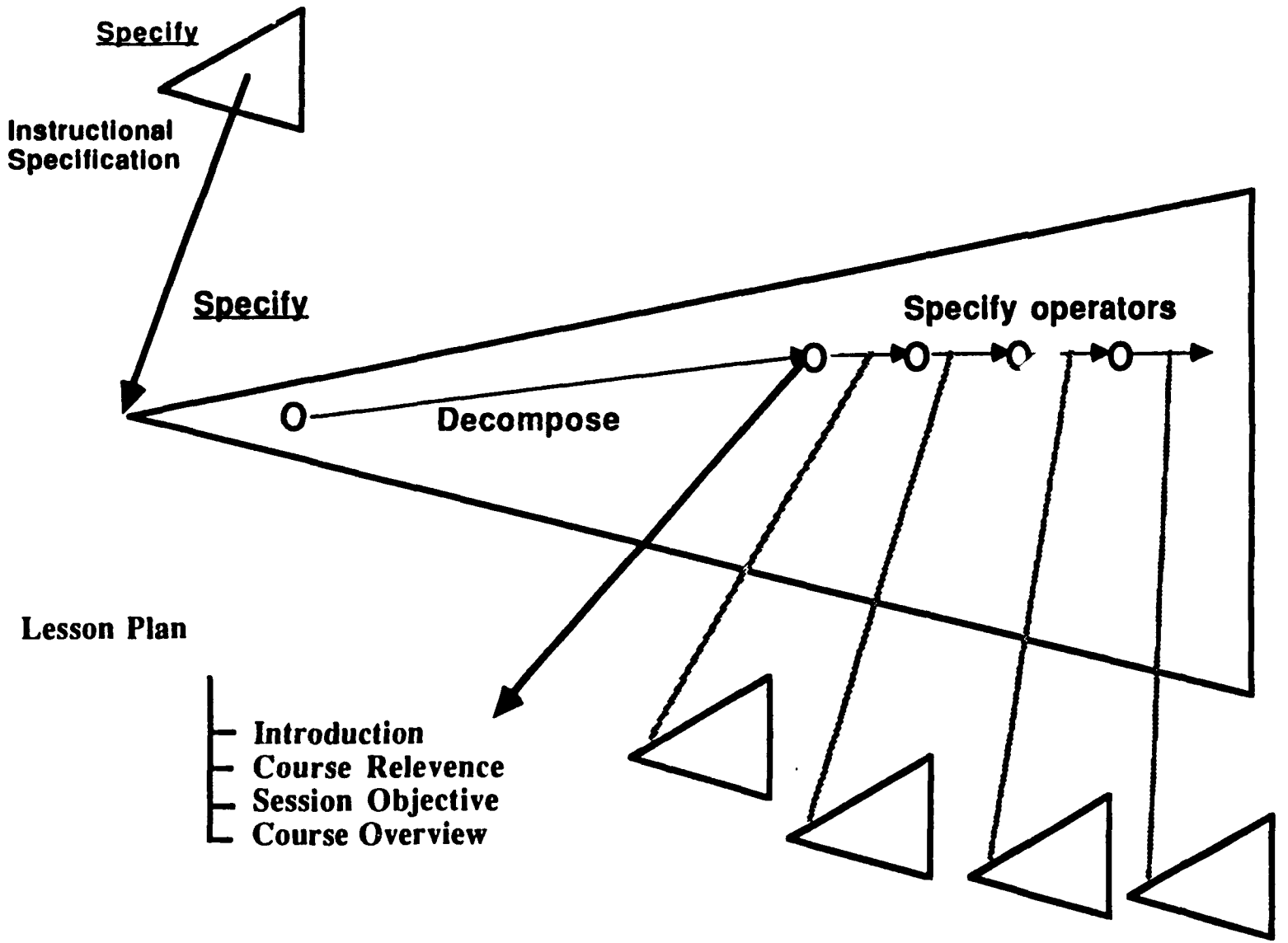

starting a session
locate & activate docs
create & edit
ending a session
destop mess-area
desktop
desktop aux-menu
directory
directory structure
copying objects
icons
move,copy del, props
icon manip
non dir Icons
windows
selecting text
adjusting selections
property sheets
properties
prop sheet commands
formatting text
option sheets
cursor manipulation
pointer shapes
evaluating ViewPoint
stats & facts
deleting files
wastebasket

Figure 9.

Figure 10.


Specify
Instructional Specification
Specify
Specify operators
Decompose
Lesson Plan
Introduction
Course Relevence
Session Objective
Course Overview

ATTACHMENT 4

JULY 1992 REPORT NO. DPS-7

The Cognitive Role of Sketching in Problem Solving

Vinod Goel

University of California at Berkeley

Portions of this research were funded by the Cognitive Science Program, Office of Naval Research under contract N00014-88-K-0233, Contract Authority Identification Number NR4422550. Reproduction in whole or part is permitted for any purpose of the United States Government. Approved for public release; distribution unlimited.

The Cognitive Role of Sketching in Problem Solving

Vinod Goel
University of California, Berkeley
goel@cogsci.berkeley.edu

Running head: Cognitive Role of Sketching

Abstract

While cognitive science has been concerned with the distinction between "diagrammatic" and "sentence-like" representations, it has not concerned itself with the distinction between different types of "diagrammatic" representations. It is argued here that even though both free-hand sketches and rigid, box-like, diagrams are "diagrammatic" representations, they differ in their degree of "density" and ambiguity and thus serve different cognitive functions. A protocol study involving nine expert designers is described. The external symbol systems subjects are allowed to use are manipulated along the dimensions of sketch-type and drafting-type diagrams. It is hypothesized that this manipulation will hamper exploration of the problem space and induce early crystallization of ideas. The results indicate statistically significant differences along the lines predicted. This leads to the conclusion that the symbol system of sketching -- by virtue of being "dense" and ambiguous -- correlates with creative, explorative, ill-structured phases of problem solving.

The Cognitive Role of Sketching in Problem Solving

It is common practice to differentiate between "language-like" and "picture-like" representations and to assign them different cognitive functions. In fact, there is by now a rather large body of work in cognitive science dealing with the distinctions between "picture-like" and "language-like" representations. This literature has a number subcomponents. At the most abstract level, there is the general question of how one differentiates "picture-like" representations from "sentence-like" representations. This issue has received surprisingly little attention, though there are some notable exceptions (Goodman, 1976; Haugeland, 1990; Palmer, 1978). A little more attention has been accorded the question of what it means to claim that the system of *internal* representation is "picture-like" or imagistic (Anderson, 1978; Anderson, 1983; Dennett, 1981; Goel, 1991; Goel, 1992; Kosslyn, 1981; Pylyshyn, 1981; Rey, 1981; Simon, 1972). However, most of the literature is concerned with adducing empirical evidence for either "picture-like" or "language-like" internal representations (Finke & Pinker, 1982; Kosslyn & Shwartz, 1977; Kosslyn & Shwartz, 1978; Meltzer & Shepard, 1974; Shepard, 1982; Shepard & Cooper, 1982; Shepard & Metzler, 1971).

There is also a small literature dealing with the cognitive role of *external* "language-like" and "picture-like" representations. Perhaps the most substantive work is due to Larkin and Simon (1987). They begin by distinguishing between "language-like" or "sentential" and "picture-like" or "diagrammatic" representations and then proceed to demonstrated how these differences can lead to cognitive differences due to differing control mechanisms, primitive operations,and resource allocations.

However, it is not common practice to further differentiate "picture-like" representations into finer-grained types.[1] "Picture-like" representations of all types -- e.g. paintings, photographs, free-hand sketches, schematic block diagrams, drafting diagrams, etc. -- are lumped together and thought to serve similar cognitive functions. The major goal of this paper is to argue that, just as there are distinctions to be made between sentential and diagrammatic representations, there are equally important distinctions to be made between different kinds of diagrammatic representations, and that different types of diagrammatic

---

[1]This finer-grained distinction is made in the design literature (Albarn & Smith, 1977; Laseau, 1989).

representations are correlated with different cognitive processes. The discussion will be restricted to *external* symbol systems. The relevance of the work to claims about the system of *internal* representations is explored elsewhere (Goel, 1991).

The strategy is to examine problem solving in design domains such as architecture, graphic design, and engineering. It is found that designers use several different types of diagrammatic symbol systems and that different systems are correlated with different stages of design development, involving different cognitive functions. In particular, it is noted that free-hand sketching diagrams are correlated with the creative, exploratory, ill-structured phases of design problem solving, while the rigid, box-like drafting drawings are correlated with the well-structured, detail design and specification phases.

Systems of sketching are differentiated from systems of drafting along a number of dimensions. The identification of these differentiating properties leads to the generation of certain hypotheses about why sketching should be correlated with the preliminary, explorative phase of design problem solving. A protocol study which manipulates these properties of sketching is carried out and described. The results are significant in the direction predicted. This leads to a conclusion about the cognitive role of sketching in problem solving.

Symbol Use in Design Problem Solving

Design is an excellent forum for studying human symbolic activity in much of its richness and diversity. Designers in fact are explicitly in the business of manipulating representations of the world (rather than the world itself). The input to the design process is a design brief. It is a written or verbal "document" which generally describes the client's current state of affairs, the reason these states are unsatisfactory, and what is required to make them satisfactory. The output of the design process is generally a set of contract documents, consisting of specifications and blueprints. The latter specify the artifact to be constructed while the former specify the procedure for construction. Thus both the input and the output of the design process are representations which describe and/or depict artifacts or processes; albeit with several interesting differences between them (Goel, 1991).

Designing, at some very abstract level, is the process of transforming one set of representations (the design brief) into another set of representations (the contract documents). However, not only are the inputs and outputs of the design process representations, all the

intervening transformations are also typically done on representations. This is not an accident. It is dictated by certain features of the design task environment.

Design typically occurs in situations where it is not possible or desirable to tamper with the world until the full extent and ramifications of the intervention are known in advance. After all, we only get one "run" on the world. And not only are actions irrevocable, they may have substantive costs associated with them. Thus it is not surprising to find that designers produce and manipulate representations of the artifact rather than the artifact itself. All the reasoning and decision making (including performance prediction) is done through the construction and manipulation of models of various sorts, including drawings, mockups, mathematical modeling, computer simulations, etc.

Not only do designers make extensive use of representations, these representations encompass many different symbol systems.[2] Three such symbol systems are natural language, sketching, and drafting. Moreover, these different systems are used in a particular sequence which in effect results in a correlation of symbol systems with different phases of design development. Four commonly identified development phases are problem structuring, preliminary design, refinement, and detail specification (Goel, 1991; Wade, 1977). Each phase differs with respect to the goals and cognitive processes associated with it.

Problem structuring is the process of retrieving information from long-term memory and external memory and using it to construct the problem space; i.e. to specify start states. goal states, operators, and evaluation functions. Problem structuring relies heavily on the client and design brief as a source of information, considers information at a higher level of abstraction, makes fewer commitments to decisions, and involves a higher percentage of add and propose operators (Goel, 1991).

Preliminary design is a phase where alternative solutions are generated and explored. Alternative solutions are neither numerous nor fully developed when generated. They emerge through incremental transformations of a few kernel ideas. These kernel ideas are images, fragments of solutions, etc. to *other* problems which the designer has encountered at some point in his life experience. Since these "solutions" are solutions to other problems which are being mapped onto the current problem, they are, not surprisingly, always out of context or in someway inappropriate and need to be modified to constitute solutions to the present problem (Goel, 1991).

---

[2] I use the terminology of 'representational system' and 'symbol system' interchangeably.

This generation and exploration of alternatives is facilitated by the abstract nature of information being considered, a relatively low commitment to the generated ideas, the coarseness of detail, and a large number of lateral transformations. A lateral transformation is one where movement is from one idea to a slightly different idea rather than a more detailed version of the same idea. Such transformations are necessary for the widening of the problem space and for the exploration and development of kernel ideas (Goel, 1991).

The refinement and detailing phases are more constrained and structured. Commitments are made to a particular solution and propagated through the problem space. They are characterized by the concrete nature of information being considered (mostly having to do with the function and structure of the artifact), a higher degree of commitment to generated ideas, attention to detail, and a large number of vertical transformations. A vertical transformation is one where movement is from one idea to a more detailed version of the same idea. It results in a deepening of the problem space (Goel, 1991).

Given these phases, we find that natural language is most prominent during the problem structuring phase, while sketching dominates the preliminary design phase. As one moves from preliminary design to refinement, the forms of sketching become more constrained until a full-fledged drafting system emerges during the detailing phase.

This state of affairs is depicted in Figure 1. The geometric form depicts the design problem space. Problem spaces are generally depicted as triangles (Laird,Newell, & Rosenbloom, 1986). The problem is mapped onto the single point at an apex and the expanding area between the apex and the base metaphorically represents the expansion of the search space. In the case of design problem solving, the problem will have a number of interpretations and thus will not map onto a single point, but rather several points or alternatives (a1, a2, ...). This mapping process is called problem structuring and the primary symbol system involved is natural language. Once this mapping is completed, the designer then expands the problem space by actively generating and considering a number of alternative solutions (a1, a2, a3, a4, ...). This process occurs during the preliminary design phase and is correlated with various forms of sketching. Finally, these several alternative solutions are quickly narrowed to one solution (a1) which is then developed at great length and detailed in the contract documents by some system of drafting. This constitutes the refinement and detailing phases and is depicted by the narrowing of the base of the geometric form in Figure 1 to an apex.

Insert Figure 1 approximately here.

Figure 2 illustrates the development of a floor plan by an architectural subject (Goel, 1991). Notice transformation of drawing types from Figure 2a to 2f as the problem space is traversed. There is an increase in the degree of explicitness and detailing in the lines, along with an introduction of dimensions, labels, and iconic symbols. (Figure 2f is actually several drawings removed from Figure 2e, and was not produced by the subject as part of the session. It is a floor plan, and is included to give the reader an indication of what the subject's final drawings would have looked like, if time had permited the production of drafting type drawings.)

Insert Figure 2 approximately here

It is important to realize that the differences in the drawings as one proceeds from Figure 2a to 2f do constitute a difference in drawing types. Sketches are not just quickly drawn, "sloppy" diagrams. Similarly, drafting drawings are not just "neat" diagrams drawn with the mechanical aids. The two constitute very different symbol systems,[3] that is, they differ in syntactic and semantic properties. Following Goodman (1976), I want to suggest that syntactically, the system of sketching differs from system of drafting in terms of the demarcation of syntactic types (or characters or symbols) and in terms of the ordering of syntactic types (or characters or symbols).[4] Semantically, the two differ in terms of the reference link, and in terms of the demarcation and ordering of reference-classes.[5] These

---

[3]A symbol system consists of a series of marks (or tokens), which instantiate types (or belong to characters/symbols), which refer to classes of objects, events, and states of affairs in the world.

[4]When talking about the syntactic elements of a symbol system I, following Goodman (1976) prefer the vocabulary of 'marks' and 'characters'. The characters are the types and should be thought of as equivalence-classes of marks, where marks are tokens. The reader may substitute the 'type'/'token' vocabulary as necessary. Also, a character is here to be more broadly construed than an element of the alphabet. A character is taken to be any symbolic expression, whether simple or complex. It can be anything from a single mark, (as in the case of a letter of the alphabet), an utterance, a gesture, etc. to -- in the limiting case -- the whole work (e.g. complete paintings, sketches, novels, plays, etc.).

[5]A reference-class (Goodman, 1976) is just the class of objects, events, states of affairs, etc. to which a syntactic type, character or symbol refers. Where one was interested in contents, instead of reference, one might speak of "content-classes".

differences are summarized in Table 1 and briefly discussed below. The reader is referred to Goodman (1976), Elgin (1983), and Goel (1992) for further discussion.

Table 1 approximately here

Demarcation of Characters: The first syntactic property along which sketching systems differ from drafting systems is the demarcation of characters (or symbols). The issue here is one of syntactic type identity. In certain symbol systems the equivalence-classes of tokens (or marks) which constitute the syntactic types are allowed to intersect, while in other symbol systems such classes are required to be disjoint. What this means is that, in an intersecting system, each token/mark will instantiate many types/characters whereas in a disjoint system each token/mark will instantiate at most one type/character. Intersection is construed broadly here to also incorporate the notion of inclusion. As such, it is just the failure of disjointness (i.e. nondisjointness).

Sketches belong to syntactically nondisjoint systems. Each token sketch may belong to many types at the same time. That is, in the absence of any agreement as to the constitutive versus contingent properties of marks, there may be no fact of the matter as to which equivalence-class or type they belong to. Thus for example, what equivalence-class or type does drawing 2 in Figure 3 (see Informal Overview of Data section) belong to? Do drawings 2 and 7 belong to the same equivalence-class or type? There may be no agreed upon answers to these questions.

Drafting drawings, in contrast, belong to a syntactically disjoint system. There is agreement as to the criteria for membership in the various equivalence-classes, and these criteria are non-overlapping. For example, the marks belonging to the symbol for door are distinct from the marks belonging to the symbol for window.

Ordering of Characters: The second property along which systems of sketching and drafting and differ is in terms of the ordering of syntactic types (or characters). The dimension of interest here is dense vs. non-dense. A symbol system is syntactically dense if the types/characters are ordered such that between any two types/characters there is a third type/character. Thus for example, the system of rational numbers (as normally ordered) is dense, whereas the system of integers and the English alphabet are not. In a symbol

system which is syntactically dense it is not possible to determine which type/character a token/mark actually instantiates. In a system which is not dense, such determination is possible, at least in principle.[6]

The system of sketching is syntactically dense because it allows for an ordering of sketch types/characters such that between any two sketch types/characters there is a third. So, for example, even it we agree that the drawing 2 in Figure 3 does in fact only belong to one equivalence-class or type, it may not be possible to specify which of several closely ordered classes it does or does not belong to.

Systems of drafting are non-dense. In a drafting system the drawings indicate only (and very roughly) relative size, shape, and location. Line lengths are conventionally restricted to preset tolerances (e.g. 1/16th of an inch) and each drawing is clearly marked with the warning: "Do not scale". Thus it is always possible to determine which type/character a token/mark belongs to.

Reference Link: Ambiguity and unambiguity characterize the relationship between a syntactic type/character and what it refers to. A syntactic type/character is ambiguous if it has different referents in the different contexts in which it appears. A syntactic type/character is unambiguous if it has the same referent in each and every context in which it appears. Ambiguity is simply the failure of unambiguity.

The system of sketching is ambiguous because sketches do not have the same referent in each and every context in which they appear. For example, does drawing 7 in Figure 3 represent a human head or a light bulb? It received both interpretations under different contexts. Drafting drawings generally have an unambiguous interpretation. For example, the symbol for a watercloset always refers to a watercloset, irrespective of context.

Demarcation of Reference-Classes: The second semantic property along which the systems of sketching and drafting differ is in the demarcation of reference-classes. As in the syntactic case, the options here are nondisjointness and disjointness. The difference is that here these labels predicate over reference-classes rather than characters. Semantic nondisjointness allows reference-classes to intersect, while disjointness prohibits such intersection and inclusion. As in the syntactic case, nondisjointness is simply the failure of disjointness.

---

[6]Things are actually more complex than this (Goodman, 1976).

The system of sketching allows for nondisjoint reference-classes. So for example, the human figure referred to by drawing 3 in Figure 3 may belong to the class of humans and the class of students. Drafting systems generally do not allow intersection or inclusion of reference-classes. For instance, the system of drafting used by architects has a symbol which denotes the class of waterclosets. It does not have any other symbol whose denotation intersects with the class of waterclosets. (The reference-class of a symbol for bathroom fixtures would include the reference-class for watercloset, but such a symbol does not exist.)

Ordering of Reference-Classes: The last differentiating dimension we will consider is the ordering of reference-classes. Again, as in the syntactic case, the options are dense and non-dense ordering, but here they predicate over reference-classes as opposed to characters. A symbol system is semantically dense if its reference-classes are ordered such that between any two classes there is a third class. A symbol system is non-dense if there are gaps between the reference-classes.

The system of sketching allows for a dense ordering of reference-classes. For example, in a perspective drawing of a human figure, every height of marks would correspond to a different class of heights of human figures in the world, and these classes of heights would course be densely ordered. In such a case it would not be possible to tell which of several sketch types a particular human height belongs to. In drafting drawings, not only the line lengths, but also the corresponding referents are conventionally restricted to preset tolerances, thus allowing for a non-dense ordering of reference-classes.

It is important to note that the system of sketching and drafting not only differ with respect to demarcation and ordering of characters and reference-classes, and the structure of the reference link, they in each case fall on the opposite extreme of the same dimensions. So where one is dense, the other is nondense. Where one is disjoint, the other nondisjoint. And while one is ambiguous, the other is unambiguous.

Given that design problem spaces involve several phases which differ in cognitive processes, and utilize several different symbol systems which correlate with cognitive processes, the question naturally arises as to why these correlations between design phases and symbol systems should occur? More particularly, as this work is restricted to understanding the cognitive role of sketching and how it differs from the role of other diagrammatic systems such as drafting, the question to be asked and answered is "why should the symbol system of sketching be correlated with preliminary design?".

Taking seriously both, the characterization of preliminary design -- as a process of creative, ill-structured problem solving, where the generation and exploration of alternatives is facilitated through a coarseness of detail, a low commitment to ideas and a large number of lateral transformations -- and the properties of the symbol system of sketching, one might propose an answer along the following lines to the above question:

1) The nondisjointness of characters in the symbol system of sketching gives the marks a degree of coarseness by allowing marks to belong to many different characters. This allows the designer to remain non-committal about the character.

2) The dense ordering of characters in the symbol system of sketching gives the marks a degree of fine-grainedness by making every distinction count as a different character. This reduction in distance between characters helps insure that possibilities are not excluded and facilitates the transformation from one character to another.

3) Ambiguity of the symbol system of sketching insures that the extensions of characters during the early phases of design are indeterminate. Ambiguity is important because one does not want to crystalize ideas too early and freeze design development.

4) The nondisjointness of reference-classes in of the symbol system of sketching results in an over-lap of the reference-classes. This overlap is needed to remain noncommittal about the exact referent of a character.

5) The dense ordering of reference-classes in the symbol system of sketching insures that possibilities are not excluded and facilitates the transformation of one idea to another.

These are possible answers to the question "why should the symbol system of sketching be correlated with the preliminary phases of design problem solving?" The general claim is that certain cognitive processes need to occur during the preliminary phase of design problem solving, and that certain properties of the symbol system of sketching facilitate these cognitive processes. A rigid, box-like symbol system such as drafting, which differs from sketching in being disjoint, nondense, and unambiguous, will hamper the relevant cognitive processes. An experiment was conducted to investigate this proposal. It is described in the next section.

Methodology and Data Base

The experimental design requires expert designers to engage in two (one hour) problem solving sessions while the (external) symbol systems they are allowed to use are manipulated along the syntactic and semantic dimensions noted in Table 1. In one condition subjects are allowed to use the symbol system of sketching, while in the other case they are requested to use a symbol system with the syntactic and semantic properties of drafting systems. The goal of the manipulation is to enable a conclusion about the impact of the syntactic and semantic properties of sketching on the design problem space. The expectation is that the absence of these properties will hamper lateral transformations and freeze design development. The hypotheses are discussed and stated more explicitly in the next section.

Subjects: Nine subjects employed as professional designers by a single multi-national corporation volunteered to participated in the study. While their professional experience varied from two to twenty years, each was deemed to be an "expert" by virtue of being trained as a designer and successfully earning a living at it. They were accepted for the study on the basis of three criteria: (i) that they had received training in the traditional drawing tools and methods of their profession, (ii) that they regularly use a computational system for some part of their design activity, and (iii) that they be familiar with the MacDraw[7] drawing package.

Six of the subjects were graphic designers; three were industrial designers. The basic difference between the two groups was that the graphic designers generally worked on tasks involving 2-D graphical/textual layout projects (such as corporate logos, posters, brochures, etc.), while the industrial designers worked on 3-D projects (such as "product shells"). Accordingly, the tasks given the two groups varied along these dimensions.

Task Descriptions: There were three graphic design tasks and two industrial design tasks. The graphic tasks required the design of (i) a poster and/or technical report cover for the new cognitive science program at UC-Berkeley; (ii) a poster to promote the Shakespeare Festival at Stratford-on-Avon in Canada; and (iii) a poster to attract Canadian tourists to the city of San Francisco. The industrial design tasks required the design of (iv) a desk time piece to commemorate Earth Day, and (v) a toy to amuse and educate a 15-month old toddler. Each of these are open-ended, "real-world" problems.

The distribution of tasks across the two symbol systems requires some explanation. First, the reason that there are three graphic design tasks instead of two is that some of the

[7]MacDraw is a registered trademark of Apple Computer, Inc.

graphic designers actually participated in three different sessions instead of two. The third session had a different goal and involved a different computational interface. It was not considered a part of this study. Second, one of the industrial designer subject was allowed to do a graphic design task. This resulted from the fact that several of the subjects considered themselves proficient at both industrial and graphic design. Thus each subject was given a choice as to which type of task they preferred. One subject chose one of each type.

As there were only nine subjects it was not possible to divide the sequence of sessions evenly between them. Distribution was therefore determined by the flip of a coin. This resulted in six subjects doing the sketching session first, followed by the session on the non-sketching (drafting type) drawing system, and three subjects doing the session on the drafting type system first, followed by the sketching session. In the case of three of the six subjects who did the sketching session first, there was a lapse of several weeks between the sketching sessions and the drafting type system sessions.

Instructions to Subjects: The subjects were given two sets of instructions. First, they were asked to "talk aloud" during the task and encouraged to solicit additional information and/or clarification from the experimenter who assumed the role of the client. The experimenter answered any questions that arose but did not initiate questions or conversation (apart from asking "what are you thinking now?" when the subject was silent for a prolonged period). Second, since the focus of the study was on preliminary design, rather than the whole design process, subjects were requested to "generate several ideas and then to explore one or two a little bit". They were not expected to come up with a final specification.

Recording of Data: All sessions were recorded on two video cameras. One camera captured the general movements and gestures of the subject while the other was focused on the piece of paper or computer screen on which the subject was drawing and/or writing. In addition, the sessions which utilized a computer drawing package (see below) were also recorded by MediaTracks[8], a piece of software which runs in the background and maintains a record of all screen activity which can then be replayed.

[8]'MediaTracks' is a registered trade mark of Farallon Corp.

Manipulation of Symbol Systems

The manipulation of symbol systems required finding a symbol system which differed from sketching only with respect to the syntactic and semantic properties identified in Table 1. So importantly, it needed to be a "drawing system" of some sort as opposed to a "discursive language". The symbol system which comes closest to meeting our needs is the system of drafting. It is -- as has been already noted above -- a subset of the system of drawing which differs from the system of sketching in just the right way. That is to say, it actually falls on the opposite extremes of the syntactic and semantic properties of interest.

In some trial studies an attempt was made to impose the discipline of drafting on the subjects during the preliminary design phase when they would normally sketch free-hand. They simply could not (or would not) follow the instructions. Failing this, the manipulation of symbol systems was effected through the manipulation of drawing tools and media.

While it is true that there is a logical distinction to be made between drawing tools and media on the one hand and symbol systems on the other, in practice this distinction is often collapsed, especially in the case of computational interfaces. A computational interface provides not only a tool and a medium for drawing, it also specifies a symbol system by easily facilitating certain marks and operations and discouraging or even disallowing others.

In one session each designer was allowed to use the tools, media, and symbol systems of his/her choice. They invariably chose to use paper and pencils and did a lot of sketching. In the second session they were requested to use a computational interface. Specifically, they were asked to use a subset of the drawing package MacDraw (version 1.9.5; with the freehand tool turned off and the grid turned on) running on a Mac II[9] with a large two-page monitor.

MacDraw, or even computational interfaces in general, are not the focus of this experiment. MacDraw simply provides a drawing tool which allows subjects to make certain types of marks and prevents them from making other types of marks. In particular, it is not a sketching tool; it implements a restrictive subset of a drawing system, not unlike the subset found in the system of drafting. The major difference between the two is that in the case of drafting, the subset is specified by convention, while in MacDraw it is enforced by the tool.

---

[9]Mac II is a registered trademark of Apple Computer, Inc.

The utility of MacDraw for our purposes lies in the fact that the symbol system it specifies differs from the system of sketching along the dimensions of interest to us (Table 1).

With the freehand tool turned off, MacDraw supports three closed figure tools (ellipse, polygon, and rectangle with rounded corners), three straight line tools, and one curved line tool, all in four different line weights. Accuracy of line lengths, angles, and locations of the marks is limited to 1/8 inch by the grid. Within these constraints one can -- indeed one must -- make marks of only particular shapes and sizes and of the utmost precision and certainty. Yet despite these limitations, it is possible to build up surprisingly complex shapes and figures (as will be seen shortly).

Given these constraints it is reasonable to individuate equivalence-classes of marks (or characters) in terms of congruence and identical orientation.[10] This leads to disjoint nonunit equivalence-classes (because the system facilitates the generation of congruent marks of identical orientation). In addition, given two arbitrary shapes/marks it is perceptually possible (for the human visual system) to determine whether they belong to the same equivalence-class. Thus the system does satisfy the two syntactic constraints of drafting type symbol systems.

The determination whether the system satisfies the two semantic constraints is more difficult. If the correlation of characters is with the corresponding geometric forms, then the unambiguity and semantic nondensity criteria are met. If however, the correlation is with arbitrary (existing or desired) states of affairs in the world, then neither unambiguity non nondensity can be guaranteed. One might think that a system which satisfies the syntactic properties will also satisfy the semantic properties (even though the two are logically independent) -- after all, how would a nondense ordering of characters pick out densely ordered reference-class? However, natural language has solved this problem with the introduction of combinatorial syntax and semantics and provides a persuasive counter example. Thus the semantic properties of a symbol system depend on the richness of the syntactic scheme and on how the subject chooses to carve up the world with it. This however will not be a problem for our investigation. An empirical measure of the syntactic and

[10]The reader will recognize that demarcating equivalence-classes in terms of congruence and identical orientation leads to the system of Oriented Geometry. The only permitted transformation is translation and members of equivalence classes share such invariant properties as angle values, line lengths, number of sides, separation of plane surface into interior and exterior, perimeter length, area enclosed by perimeter, and orientation with respect to some axis.

semantic properties of both symbol systems, as they are actually by the subjects during the experiment, will be provided.

In summary, there are a number of positive reasons for using MacDraw to provide the alternative symbol system to free-hand sketching. MacDraw provides considerable drawing power -- at least sufficient to do simple drafting-type drawings -- while differing from the system of sketching along the relevant syntactic and semantic dimensions. It also preserves some of the basic spatial properties exemplified by the system of sketching.[11] It is something that the subjects were familiar or proficient with,[12] and it is readily available. There is, however, one concern about using a computational interface which should be voiced. Paper and pencil and computational systems currently require very different physical actions to create marks. Dragging and clicking a mouse has a very different "feel" to it than drawing on paper with a lead pencil. I am assuming this is an inconsequential difference once a certain degree of competence has been acquired with a mouse (as each of our subjects had), or with a pencil for that matter of fact.

Informal Overview of Data

Before specifying the coding scheme and actually offering an analyses of the data, it may be a good idea to get an intuitive feel for the behavioral and design output differences across the two conditions. Figure 3 and Figure 4 reproduce the output of two design sessions, one free-hand and the other MacDraw. The free-hand drawings (Figure 3) are from a Cognitive Science Program Poster task while the MacDraw drawings (Figure 4) are from a Shakespeare Festival Poster session.

Figure 3 about here

Figure 4 about here

---

[11] While it does preserve the gross spatial properties (e.g. shape, size, location, etc.) it does not preserve the more subtle properties, or most of the expressive properties of the system of sketching. At least one subject complained about this.

[12] The subjects all used sophisticated computational drawing systems as part of their jobs. These systems for the most part were a superset of MacDraw.

Informally, and very briefly, the difference between the two cases seems to be the following: In free-hand sketching, when a new idea is generated, a number of variations of it quickly follow. The variations expand the problem space and are necessary for the reasons noted earlier. One actually gets the sense that the exploration and transformation of ideas is happening on the paper in front of one's eyes as the subject moves from sketch to sketch. Indeed, designers have very strong intuitions to this effect.

When a new idea is generated in MacDraw, its external representation (in MacDraw) serves to fixate and stifle further exploration. Most subsequent effort after the initial generation is devoted to either detailing and refining the same idea or generating the next idea. One gets the feeling that all the work is being done internally and recorded after the fact, presumably because the external symbol system cannot support such operations.

These observations are reinforced by some of the verbalizations of the subjects. Here are a few fragments to give one the flavor of their utterances:

> Something happens in the sloppy interaction when pen hits paper which is not happening here.
>
> You almost get committed to something before you know whether you like it or not.
>
> I have to decide before hand what I want before I can draw it.
>
> It is very frustrating.... unless you have an idea of what you want before you even sit down.
>
> Sketching is noncommittal. This is forcing me to make a commitment.
>
> I just sketch a shape and look at it and have something come out of it.
>
> I feel like I am doing a final design for something that I already know....
>
> Its hard to play on here. Here you have to have an idea and do in and do it.
>
> In sketching you can start drawing a line and it will turn into something else. With this you have to know what you are doing.

Their comments are not confined to complaining about MacDraw. At a certain stage in the design process they considered computational systems to be of value, and even asked for them during the latter stages of their free-hand sessions.

> I really can't go any further. I would now move to the computer to clean this up and get the lines and types right....
>
> At this point it would be nice to switch over to something like Adobe and to have access to all the typefaces and fonts and libraries of pictures....

Before developing a coding scheme and examining the data more rigorously, it is worth being very explicit about what we are looking for. This is articulated in the next section

Hypotheses

There are two sets of questions which the data need to address. The first set has to do with whether the manipulation of symbol systems is actually successful in varying the relevant syntactic and semantic properties across the two conditions. The second has to do with the impact of this manipulation on the design problem space.

Arguments have already been offered as to why MacDraw and sketching -- viewed in the abstract -- should fall on opposite dimensions of the syntactic properties in Table 1. That is, if symbol systems are defined intensionally -- in terms of the characters and reference-classes allowed for by the "rules" of the system, then free-hand sketching turns out to be syntactically and semantically nondisjoint and dense, and ambiguous, while MacDraw turns out to be, at least, syntactically disjoint and nondense. But the question remains, do these symbol systems -- *as they are in fact used by our subjects during the experiment* -- differ with respect to the syntactic (and semantic) properties in the manner claimed? An empirical measure of this would be desirable.

This calls for an extensional definition of symbol systems, where a symbol system is defined not in terms of the *possible* characters and reference-classes that can be generated by the rules, but just in terms of the *actual* characters and reference-classes generated by the subject during the experiment. On the extensional characterization we can get (relative) empirical measures of density and ambiguity and/or nondisjointness. The claim that free-hand sketching and MacDraw, defined extensionally, do differ with respect to the relevant syntactic and properties can be captured in the following three hypotheses:

H1) Free-hand sketching is syntactically more dense than MacDraw.

H2) Free-hand sketching is semantically more dense than MacDraw.

H3) Free-hand sketching is more ambiguous and/or more nondisjoint than MacDraw.

There are several important differences between the formulation of these hypotheses and the initial characterization of sketching and drafting systems. One difference is that in formulating these empirical hypotheses the logically distinct notions of nondisjointness and unambiguity have been collapsed. This I suspect is an artifact of the particular methodology employed. It was not sufficiently fine-grained to allow for a differentiation between nondisjointness and unambiguity. Presumably a different experiment design could overcome this problem.

Another difference is that the notions of disjointness, density, and ambiguity are really binary criteria. In the above hypotheses they have been reduced to relative concepts. The question is no longer whether MacDraw strictly satisfies the requirements of disjointness, nondensity, and unambiguity, and free-hand sketching strictly satisfies the requirements of nondisjointness, density, and ambiguity, but rather, which ends of the spectrum they roughly fall on. There are two reasons for this shift. First, on a purely extensional definition, a symbol system cannot of course be dense in the technical sense. Given a finite number of characters and reference-classes there will be no ordering of them which is dense. One can however use the notion of density in a less technical sense and show that the characters and reference-classes of one system are more closely (or "densely") ordered than that of another system. It is in this sense that the term 'density' is used in the balance of this article.

The second reason for the shift from an absolute to relative vocabulary is that we are confronting the general problem of maintaining black and white distinctions in a world consisting of various shades of grey. This is a general difficulty in mapping logical concepts onto the psychological domain. There are no easy solutions or explanations.

Once it is verified that the two symbol systems, free-hand sketching and MacDraw, are being used in the desired way, one can proceed to investigate the impact of this manipulation on problem solving behavior. The prediction here is that syntactic and semantic density will facilitate lateral transformations by allowing for closely ordered (finely individuated) characters and referents and/or contents.[13] In addition, ambiguity and/or nondisjointness of characters will facilitate lateral transformations by allowing for multiple interpretations and/or overlapping characters. This leads to the following hypothesis:

[13]In the coding scheme will actually utilize the notion of contents or ideas rather than referents.

H4) Symbol systems which are nondense, unambiguous and/or disjoint hamper the exploration and development of alternative solutions (i.e. lateral transformations) and force early crystallization.

(Notice that the design of the experiment is such that it will not be possible to determine what percentage of lateral transformations are due to density versus ambiguity and/or nondisjointness.)

To investigate these hypotheses we need a quantitative measure of syntactic density (H1), semantic density (H2), ambiguity and/or nondisjointness (H3), and the number of lateral transformations (H4) across the two conditions. A coding scheme was developed to facilitate just these measures. It is described in the next section.

Coding Scheme

The protocols were first broken up into episodes along the lines of alternative solutions, a notion derived from Goel (1991). There is a one-to-one correspondence between episodes and alternative solutions. Episodes were then correlated with the sketches and drawings generated during the sessions. A one-to-one correlation was also desired between drawings and alternative solutions. The first step in setting up this correlation was to discount any episodes/alternative solutions which did not result in marks-on-paper. The rationale for this is simply that the lack of marks-on-paper indicates a lack of commitment, and more importantly, it is after all these marks which are the object of study. The second step is to individuate these marks into drawings and sketches.

The individuation of drawings and sketches along these lines was reasonably straight forward in the case of the free-hand sessions. For the most part, the individuation was done by the subject, by drawing a rectangle around each separate drawing and even numbering them. (The numbering was done at the experimenter's request.) Most subjects also assigned a linguistic label to each drawing which also helped in the individuation process. This individuation of sketches resulted in a unique correspondence to alternative solutions (Figure 5).

Figure 5 Approx here

Individuating the drawing output of the MacDraw sessions so that they uniquely corresponded to alternative solutions (or episodes) was more problematic. The difficulty resulted from the fact that the ability to cut, paste, delete, move, resize, translate, etc. meant that changes were often made in place rather than resulting in a new drawing. This of course destroys the one-to-one correspondence between drawing output and alternative solutions. The strategy used to overcome this difficulty was to individuate MacDraw drawing output not only spatially, but also temporally. This temporal individuation did provide a unique mapping between alternative solutions and drawing output.

This procedure results in a similar number of drawings or episodes across the two conditions, and the episodes turn out to have similar temporal duration (see Table 2, in the Results section). The temporal duration of an episode was determined by the time it took the subject to complete the accompanying drawing. The clock was started when the drawing was started and stopped when the drawing was completed or abandoned. Any periods of drawing inactivity between the start and completion of drawings were counted as part of the episode. The fact that there are no significant differences in the number and duration of episodes across the two conditions suggests that similar size "chunks" are being individuated.

No finer grained breakdown of the protocol was undertaken. All the subsequent coding and analysis is carried out at the episode or alternative solution, or drawing level. This is appropriate because our hypothesized differences across the two conditions are stated at the level of alternative solutions and the properties of symbol systems we are interested in are to be found at the level of complete drawings (actually in the relationship between drawings), rather than the internal structure of drawings (Goel, 1992).

Given this initial breakdown of the protocols, the drawings accompanying each episode were coded along the following three dimensions: (i) *source type:* where did the diagram originate from?; (ii) *transformation type:* how was it generated?; and (iii) *number of reinterpretations:* did the drawings undergo any reinterpretations? The scheme is formally stated in Backus-Naur form in Figure 6 and discussed below.

Figure 6 Approx here

The *source type* category is applied at both a syntactic level and a semantic level. So every drawing has a *syntactic source* and a *semantic source*. The syntactic source traced the origin of the drawing while the semantic source traced the origin of the idea or content of the drawing. There is no *a priori* reason that these two components should coincide and often did not. The two possibilities for a source are *long-term memory (LTM)* and *previous solutions*.

Categorizing a drawing as originating from LTM at the syntactic level is to say that the equivalence-class to which the marks belong (i.e. syntactic type/character) is new. Categorizing a drawing as originating from LTM at the semantic level is to say that the idea or content of the drawing is new. In either case it is just another way of saying we don't know where the drawing or idea came from but that it is not related to previously generated drawings or ideas in any obvious way. The first drawing in every session is automatically classified as originating from LTM (at both the syntactic and semantic levels), as are subsequent drawings which have no obvious relations to previous drawings. For example, drawings 1 and 5 in Figure 3 are classified as originating from LTM at both the syntactic and semantic levels. Drawing 7 in Figure 3 is classified as originating from LTM at the semantic level but not the syntactic level. (It is syntactically related to previous drawings but has received a new interpretation as a "light bulb".) In Figure 4 drawings 1, 8, 10, and 11 are classified as originating from LTM at both the syntactic and semantic levels. Drawing 6 (Figure 4) is classified as originating from LTM at the syntactic level but not the semantic level (it is the same idea as in the previous drawing, that of portraying a Shakespearean actor in period costume). The LTM category serves as a measure of "new generations" or new solutions.

Where a drawing is categorized as originating from one or more *previous solutions*, it means that it is related to one or more existing drawings (which correspond to previous solutions in a one-to-one manner) by the relation of *variation, identical+/-*, or *identical.*

A *variation* rating means that the current drawing is recognizably similar to earlier drawings. At the syntactic level this means that the equivalence-class of marks (i.e. syntactic types/characters) constituting the drawing are closely related to, but distinct from, the equivalence-class of marks constituting one or more previous drawings. A *variation* rating at the semantic level means that the idea or content of the drawing is similar (but not identical) to the ideas or contents of one or more previous drawings. For example, in Figure 3, drawing 2 is considered a variation of drawing 1 at both the syntactic and semantic levels. Drawing 7 (Figure 3) however, is considered a variation of drawing 1 at the syntactic level but not at the semantic level (because it has received a totally new interpretation as a "light

bulb". In Figure 4, variations are much less frequent. A good example is however provided by drawing 12, which is considered a syntactic and semantic variation of drawing 8.

The *variation* category serves as a measure of density. The connection between the two can be seen with the aid of the following example. Consider two symbol systems, *SS1* and *SS2*. In *SS1* characters consist of equivalence-classes of line lengths which, when measured in feet correspond to the integers. So we have lengths of 1', 2', 3', etc. In *SS2* characters consist of equivalence-classes of line lengths which, when measured in feet correspond to the rational numbers. So we have lengths of 1', 2', 3'...; but also lengths of 1.5', 2.5', 3.5'... and 1.25', 2.25', 3.25'... and 1.125', 2.125', 3.125'... and so on. Lines of lengths 1.125' and 1.25' are no more identical then lines of length 1' and 2', neither of these pairs belongs to the same equivalence-class. However, line lengths of 1.125' and 1.25' are much more "similar" or "closer to each other" -- with respect to length -- than lines of 1' and 2'. Thus the notions of "similarity" or "closeness" seem to be an integral (necessary?) part of density.

An *identical+/-* rating means that the drawing is identical to a previous drawing except for specifiable differences. At the syntactic level this means that the marks constituting the drawing belong to the equivalence-class of marks constituting an earlier drawing except for a few specifiable details. At the semantic level it means that the idea or content of the drawing is identical to the content of an earlier drawing except on one or two specific counts. Examples of this category are common in the MacDraw condition. For instance, in Figure 4 drawing 2 is syntactically and semantically identical+/- to drawing 1, drawing 3 is syntactically and semantically identical+/- to drawing 2, and drawing 4 is syntactically and semantically identical+/- to drawing 4. While this category can occur in the free-hand case, it is not common.

The identical+/- category importantly differs from the variation category in that what is involved is a refinement or augmentation of an existing drawing, not the generation of a different drawing. That this is the case reveals an important point about the individuation of drawings. It indicates that the individuation is not strictly on the basis of congruence and orientation as suggested earlier. In the actually coding, congruence and orientation are used as guide lines along with a host of semantic and contextual cues to determine sameness of drawing.

An *identical* rating means that the drawing is identical to a previous drawing. At the syntactic level it signifies a type identity (in terms of congruence and orientation) between the equivalence-class of marks constituting the current drawing and equivalence-class

constituting earlier drawings. At the semantic level an identical rating signifies a type identity between the content of the current drawing and the content of earlier drawings. This category never occurred in the free-hand case. Though it is important to note that both logically and practically it could have occurred, for example through tracing or photocopy of drawings. Interestingly, tracing was frequently used in the free-hand sessions. However, it never resulted in an identical drawing, but always a variation of the old drawing. The identical category did occasionally occur in the MacDraw case.

The *identical+/-* and *identical* categories are not considered to be a measure of density because they do not result in new drawings or ideas.

The *transformation type* category is also applied at both the syntactic and semantic levels. Syntactic transformations are defined over the equivalence-classes of marks constituting the drawing while semantic transformations are defined over the contents of drawings. Transformations are classified as either *new generations* or *transformations of previous solutions*.

New generations result in new drawings unrelated to previous drawings. At the syntactic level a new generation means that the equivalence-class of marks which constitute the drawing is unrelated to any earlier equivalence-classes of marks. At the semantic level it signifies that the content of the current drawing is unrelated to the content of earlier drawings. All (and only) syntactic and semantic source types which had their origin in LTM are classified as new generations. Thus for example, drawings 1 and 5 in Figure 3 are classified as new generations at both the syntactic and semantic levels. Drawing 7 in Figure 3 is classified as a new generation at the semantic level but not the syntactic level. (It is syntactically related to previous drawings but has received a new interpretation as a "light bulb".) In Figure 4 drawings 1, 8, 10, and 11 are classified as new generations at both the syntactic and semantic levels. Drawing 6 (Figure 4) is classified as a new generation at the syntactic level but not the semantic level (it is the same idea as in the previous drawing, that of portraying a Shakespearean actor in period costume).

Current solutions or drawings are often transformations of previous solutions or drawings. Three such transformations are recognized, *lateral transformations*, *vertical transformations*, and *duplication*. The concepts of lateral and vertical transformations have already been introduced. To repeat, a *lateral transformation* modifies a drawing into another related but distinctly different drawing (as opposed to a more detailed version of the same drawing, a totally unrelated drawing, or an identical drawing). A *vertical transformation*

reiterates and reinforces an existing drawing through explication and detailing. A *duplication transformation* results in movement from a drawing to a type identical drawing. Syntactic transformations relate equivalence-classes of marks which constitute drawings while semantic transformations relate the contents.

Lateral transformations are very common in the free-hand sessions. To take just one example, the transformation from drawing 9 to drawing 10 in Figure 3 is a lateral transformation at both the syntactic and semantic levels. In Figure 4, the best example of a lateral transformation is the transformation from drawing 8 to drawing 12. Vertical transformations of the other hand are more common in MacDraw. Good examples from Figure 4 are the transformations from drawing 1 to 2, 2 to 3, and 3 to 4. Duplicate transformations only occurred in Macdraw.

Finally, *reinterpretation* occurs where a subject assigns one meaning to a drawing, and then immediately, or at some later point in the session, assigns a different meaning to the same drawing. For example, drawing 1 in Figure 3 was a "silhouette figure going from dark to light". It was later reinterpreted as a "light bulb" (drawing 7) and a "Berkeley student" (drawing 12). In Figure 4, the "Shakespearean actor" in drawing 5 was reinterpreted as Shakespeare in drawing 7. Reinterpretation provides a measure of ambiguity and/or nondisjointness. As already noted, this experimental design does not permit a differentiation between these two notions.

It is essential to note that the categories involved in the measurements of syntactic density (hypothesis H1), semantic density (hypothesis H2), and ambiguity and/or nondisjointness (hypothesis H3) are logically independent of each other. They are also independent of the measurement of lateral and vertical transformations (hypothesis H4). This latter point is particularly important. If there was a conceptual connection between the categories of the two sets of hypotheses then the measure of a successful manipulation of the relevant syntactic and semantic properties across symbol systems would logically imply a confirmation of the hypothesis H4.

Results

It is now time to compare the problem spaces across the two conditions. The first thing to note is that there were no statistically significant differences in the mean durations of episodes, the mean number of episodes per session or the mean duration of sessions (Table

2). This suggests that the subjects did not have a great deal of difficulty in using MacDraw. They were neither so frustrated as to shorten the sessions nor so handicapped as to be unable to generate episodes or drawings. There were however significant differences across the two conditions in the types of episodes and their accompanying transformations. These data are discussed below and summarized in Table 3.

Table 2 approximately here

First of all, as hypothesized, the sketches or episodes in free-hand drawing were much more densely ordered (i.e. more of them received a *variation* rating, see section on coding scheme) than the drawings or episodes in MacDraw. At the syntactic level a mean of 11.2 (S= 5.1) sketches per session received a variation rating in free-hand while in MacDraw a mean of only 3.0 (S= 3.4) drawings per session received a variation rating. The difference is statistically significant (F (1,8)=46.0, p =.0007, one-tail).

Table 3 approximately here

Second, also as hypothesized, a similar difference exists at the semantic level. In free-hand sketching a mean of 10.4 (S= 1.3) episodes per session were categorized as variations, while a mean of only 4.1 (S= 3.9) episodes per session were categorized as such in the MacDraw case. While there is a narrowing of the gap from the syntactic case, the difference is statistically significant (F (1,8)=33.6, p =.0017, one-tail). The reason for this narrowing is discussed below.

Third, also as predicted, the free-hand sketches are more ambiguous and/or nondisjoint than the MacDraw drawings. There were a mean of 2.4 (S= 3.4) reinterpretations in the free-hand sessions as compared to a mean of 0.67 (S= 0.87) in MacDraw. These figures fall on the margin of significance (F (1,8)=6.5, p =.05, one-tail).

These results allow us to reject the null hypotheses associated with hypotheses H1, H2, and H3 and suggest that the two symbol systems are indeed being used in the manner predicted. That is, the free-hand sketches belong to a symbol system which nondisjoint,

dense, and ambiguous, while MacDraw drawings belong to a symbol system which comes close to being disjoint, nondense, and unambiguous. Now that we believe that our manipulation of symbol systems via tools and media has been successful, we need to investigate whether this has the predicted impact on the design problem space.

The fourth hypothesis, H4, predicted that the replacement of sketching type symbol systems with drafting type symbol systems would hamper the exploration and development of alternative solutions (i.e. lateral transformations) and force early crystallization of the design. A comparison of the mean number of lateral transformations (both at the syntactic and semantic levels) across the two conditions shows this to be indeed the case (Table 4).

Table 4 approximately here

At the syntactic level the free-hand (sketching) condition resulted in a mean of 8.9 (S= 4.4) lateral transformations per session. This is significantly more (F (1,8)=20.8, p =.006, one-tail) than the mean of 3.2 (S= 3.2) found in the MacDraw (non-sketching) case. At the semantic level the free-hand (sketching) condition resulted in a mean of 8.0 (S= 3.3) lateral transformations. This is again significantly more (F (1,8)=15.7, p =.01, one-tail) than the mean of 3.9 (S= 3.4) found in the MacDraw (non-sketching) case.

Discussion & Conclusion

Thus the data seem to provide the evidence needed to reject the null hypothesis associated with hypothesis H4. However, before actually accepting H4, there are several alternative hypotheses that need to be considered. Hypothesis H4 predicts that the manipulation will hamper certain cognitive processes. The first difficulty is that *any* manipulation which results in a deviation from normal working conditions -- whether it be an uncomfortable room temperature or being forced to draw with a 7 pound pencil -- will hamper cognitive processes. So the differences that have been noted may have nothing to do with the theoretical reasons underlying the manipulation, but may simply result from the fact that one system of drawing (free-hand sketching) was more familiar, or easier to use, than the other (MacDraw).

There are several reasons for rejecting this alternative hypothesis. First, as already noted, the manipulation does not affect the duration of sessions, the number of episodes generated per session, and the duration of these episodes (see Table 2). Neither does it have any effect on the number of new alternative solutions generated per session. At the syntactic level subjects generated a mean of 5.2 (S= 3.9) new alternatives per session for the free-hand condition as compared with a mean of 4.0 (S= 1.6) for the MacDraw condition (F (1,8) < 1). At the semantic level the free-hand condition resulted in 5.6 (S= 3.9) new episodes while the MacDraw condition resulted in 3.9 (S= 1.2) episodes (F (1,8) < 1). So it does not seem to be the case that free-hand sketching is easier to use, or simply more familiar than the MacDraw. Second, it is interesting to note that subjects actually asked for the computational medium during the latter stages of their free-hand design sessions.

A second alternative hypothesis which may account for the results is that, the differences observed are not a result of the manipulation, but rather the protocol analysis methodology. The claim here is that there are really no differences across the two conditions. What seems like a disruption of lateral transformations can be more simply accounted for as an unfortunate artifact of the methodology. The fact of the matter is that the system of *internal* representation is nondisjoint, dense, and ambiguous. Thus there is a better match between it and the system of *sketching than* the system specified by MacDraw. This fact combined with a well accepted hypothesis about protocol analysis -- that a more complete record of internal activity will result if there is a good match between the internal and external symbol systems than if there is not (Ericsson & Simon, 1984) -- suggests that what is being interpreted as a disruption of certain cognitive processes is just a difference in the completeness of the records. That is, the free-hand protocols are a more complete record of cognitive activity than the MacDraw protocols. If the two records were equally complete (or incomplete) no difference in the number and distribution of lateral and vertical transformations across the two conditions would be apparent.

I have much sympathy for this interpretation. And there is some evidence to support it. The discrepancies between the correspondence of the various syntactic and semantic measures noted above suggest that less of the cognitive burden is being off loaded on the external symbol system in the MacDraw case than the free-hand case. One way of dealing with this interpretation is to examine and evaluate the quality of the designed artifact across the two conditions. This however, is not feasible for a number of reasons. First, given the time constraints, subjects were unable to generate a complete design. Second, it is notoriously difficult to evaluate design solutions.

However, the acceptance of this interpretation does not jeopardize the conclusion I wish to draw from the study, in fact, it leads to a much stronger conclusion. I am postulating that different external symbol systems are correlated with different cognitive functions. The cost of accepting this alternative hypothesis is to postulate that different external symbol systems are correlated with different cognitive machinery.

I have argued elsewhere (Goel, 1991) that this may indeed be the right conclusion. However, this argument cannot be made solely (or even primarily) on the basis of these results. Therefore, the conclusion to be drawn from the study seems to be that (i) there are good reasons to differentiate between different types of "picture-like" representations and (ii) free-hand sketches -- by virtue of allowing syntactic and semantic density and intersection/inclusion, and being ambiguous -- play an important role in the creative, explorative, open-ended phase of problem solving. This role includes the facilitation of lateral transformations and the prevention of early fixation or crystallization (via density, ambiguity and/or nondisjointness of drawings and contents and/or referents). These functions are hampered by external representations which lack these properties.

References

Albarn, K., & Smith, J. M. (1977). *Diagram: The Instrument of Thought.* London: Thames and Hudson.

Anderson, J. R. (1978). Arguments Concerning Representations For Mental Imagery. *Psychological Review, 85*(4), 250-277.

Anderson, J. R. (1983). *The Architecture of Cognition.* Cambridge: Harvard University Press.

Dennett, D. C. (1981). The Nature of Images and the Introspective Trap. In N. Block (Eds.), *Readings in Philosophy of Psychology, Vol. 2* London: Methuen.

Elgin, C. Z. (1983). *With Reference to Reference.* Indianapolis, IN: Hackett Publishing.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data.* Cambridge, Massachusetts: The MIT Press.

Finke, R. A., & Pinker, S. (1982). Spontaneous Imagery Scanning in Mental Extrapolation. *Journal of Experimental Psychology: Human Learning and Memory, 8*, 142-147.

Goel, V. (1991) *Sketches of Thought: A Study of the Role of Sketching in Design Problem Solving and its Implications for the Computational Theory of Mind.* Ph.D. Dissertation, University of California, Berkeley.

Goel, V. (1992). Specifying and Classifying Representational Systems: A Critique and Proposal for Cognitive Science. *Proceedings of the Conference on Cognition and Representation,* SUNY, Buffalo.

Goodman, N. (1976). *Languages of Art: An Approach to a Theory of Symbols (second edition).* Indianapolis, IN: Hackett Publishing.

Haugeland, J. (1990). Representational Genera. In W. Ramsey,D. Rumelhart, & S. Stich (Eds.), *Philosophy and Connectionist Theory.* Hillsdale, N.J.: Lawrence Erlbaum.

Kosslyn, S. M. (1981). The Medium and the Message in Mental Imagery: A Theory. *Psychological Review, 88*(1), 46-66.

Kosslyn, S. M., & Shwartz, S. P. (1977). A Simulation of Visual Imagery. *Cognitive Science, 1*, 265-295.

Kosslyn, S. M., & Shwartz, S. P. (1978). Visual Images as Spatial Representations in Active Memory. In A. R. Hanson & E. M. Riseman (Eds.), *Computer Vision Systems* N.Y.: Academic Press.

Laird, J. E.,Newell, A., & Rosenbloom, P. S. (1986). *Soar: An Architecture for General Intelligence* (Tech. Report No. CMU-CS-86-171). Dept. of Computer Science, Carnegie-Mellon University.

Larkin, J. H., & Simon, H. A. (1987). Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science, 11*, 65-99.

Laseau, P. (1989). *Graphic Thinking for Architects and Designers.* N.Y.: Van Nostrand Reinhold.

Meltzer, J., & Shepard, R. N. (1974). Mental Images and Their Transformations. In R. Solso (Eds.), *Theories in Cognitive Psychology: The Loyola Symposium* Hillsdale, N.J.: Lawrence Erlbaum Associates, Inc.

Palmer, S. E. (1978). Foundational Aspects of Cognitive Representation. In E. R. &. B. Lloyd (Eds.), *Cognition and Categorization.* Hillsdale, N.J.: Lawrence Erlbaum Assoc.

Pylyshyn, Z. W. (1981). The Imagery Debate: Analogue Media Versus Tacit Knowledge. *Psychological Review, 88*(1), 16-45.

Rey, G. (1981). Introduction: What are Mental Images? In N. Block (Eds.), *Readings in Philosophy of Psychology, Vol. 2* Cambridge, Mass.: Harvard University Press.

Shepard, R. N. (1982). Perceptual and Analogical Bases of Cognition. In E. C. T. W. J. Mehler & M. Garrett (Eds.), *Perspectives on Mental Representation* Hillsdale, N.J.: Lawrence Erlbaum.

Shepard, R. N., & Cooper, L. A. (1982). *Mental Images and their Transformations.* Cambridge, MA: MIT Press.

Shepard, R. N., & Metzler, J. (1971). Mental Rotation of Three-Dimensional Objects. *Science, 171*, 701-703.

Simon, H. A. (1972). What is Visual Imagery?: An Information Processing Interpretation. In L. W. Gregg (Eds.), *Cognition in Learning and Memory* --: John Wiley & Sons, Inc.

Wade, J. W. (1977). *Architecture, Problems and Purposes: Architectural Design as a Basic Problem-Solving Process.* N.Y.: John Wiley & Sons.

Author Notes

The author is indebted to Peter Pirolli, Steve Palmer and Jeff Shrager for advice and comments which greatly improved this paper. This article is based on a chapter of the author's dissertation (UC-Berkeley, 1991). The research and writing have been supported by a Canada Mortgage and Housing Corporation Scholarship, a Gale Scholarship, and a research internship at the Systems Sciences Lab at Xerox PARC, and CSLI, Stanford University, and an Office of Naval Research Cognitive Science Program grant to Peter Pirolli (#N00014-88-0233).

Table 1:
Differentiating Sketching and Drafting Systems

	Drafting Systems	Sketching Systems
Syntactic Properties		
Demarcation of Types	Disjoint	Nondisjoint
Ordering of Types	Non-dense	Dense
Semantic Properties		
Reference Link	Unambiguous	Ambiguous
Demarcation of Reference-Classes	Disjoint	Nondisjoint
Ordering of Reference-Classes	Non-dense	Dense

Table 2:

Mean Duration of Sessions and Episodes in Minutes, and Mean Number of Episodes per Session

	Free-hand	**MacDraw**
Mean duration of sessions (min)	56.7 (S =13.1)	53.2 (S =14.6)
Mean duration per episode (min)	2.5 (S =1.3)	2.8 (S =0.6)
Mean number of episodes	16.4 (S =7.8)	14.4 (S =6.1)

Table 3:
Mean Numbers of Densely Ordered Episodes and
Reinterpreted Episodes per Session

	Free-hand	**MacDraw**
Syntactic Density	11.2 (S=5.1)	3.0 (S=3.4)
Semantic Density	10.4 (S=1.3)	4.1 (S=3.9)
Ambiguity	2.4 (S=3.4)	0.67 (S=0.87)

Table 4:

Mean Numbers of Lateral Transformations per Session

	Free-hand	**MacDraw**
Syntactic Lateral Transformations	8.9 (S=4.4)	3.2 (S=3.2)
Semantic Lateral Transformations	8.0 (S=3.3)	3.9 (S=3.4)

Figure Captions

Figure 1. Correlation of symbol systems and design phases. See text.

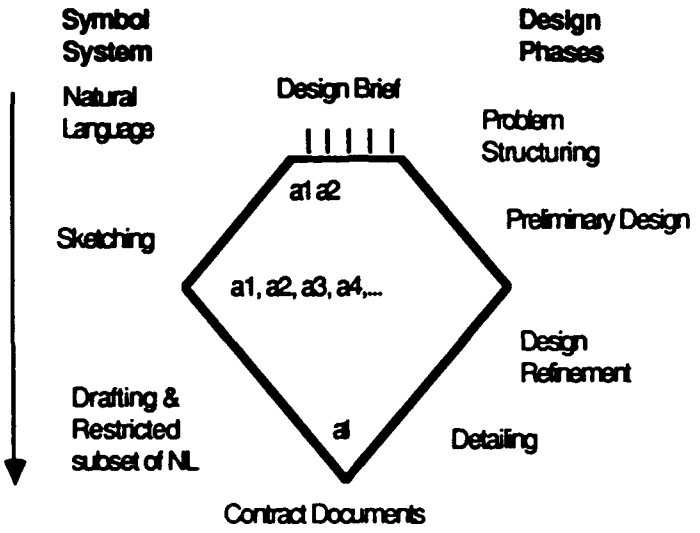
Figure 2. Transformations in external drawings accompanying design problem solving phases.

Figure 3. Output of a sketching session from a Cognitive Science Program Poster task.

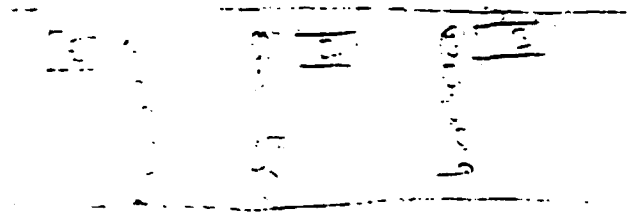
Figure 4. Output of a MacDraw session from a Shakespeare Festival Poster task.

Figure 5. There is a one-to-one correspondence between alternative solutions, episodes & drawings.

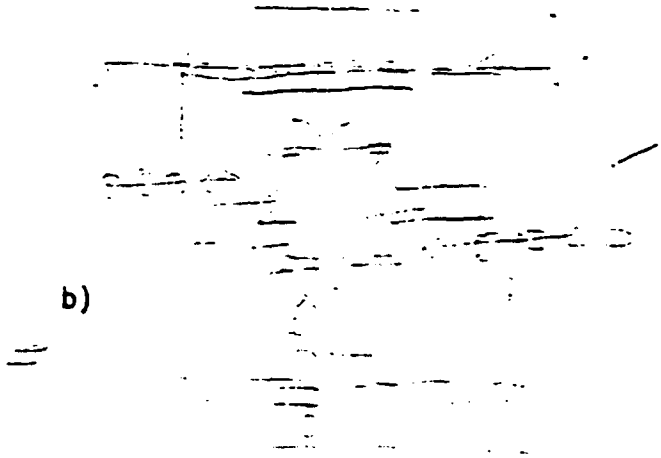
Figure 6. Coding scheme in Backus-Naur form.


Symbol System
Design Phases
Natural Language
Design Brief
Problem Structuring
a1 a2
Preliminary Design
Sketching
a1, a2, a3, a4,...
Design Refinement
Drafting & Restricted subset of NL
a1
Detailing
Contract Documents

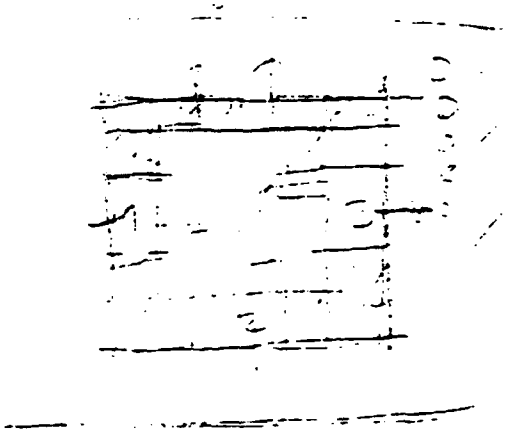
a)



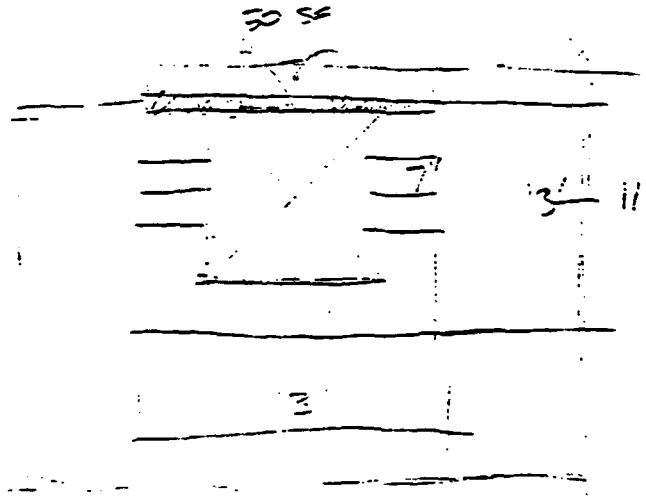
b)



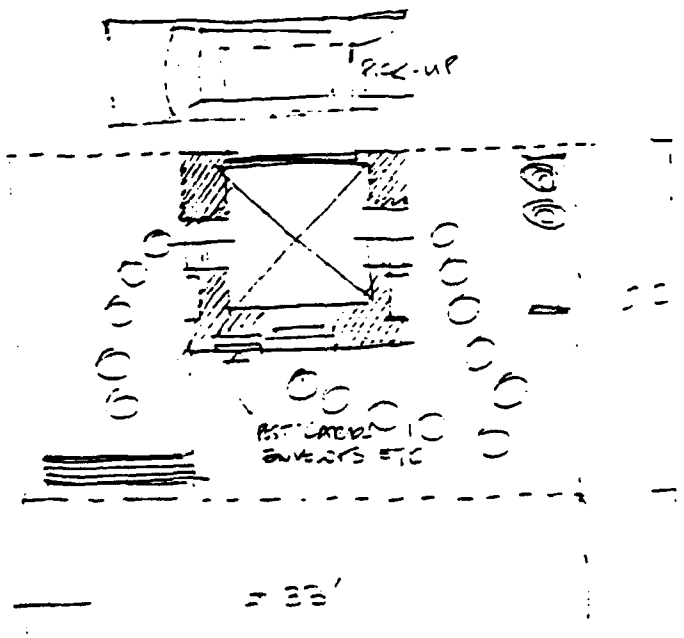
c)



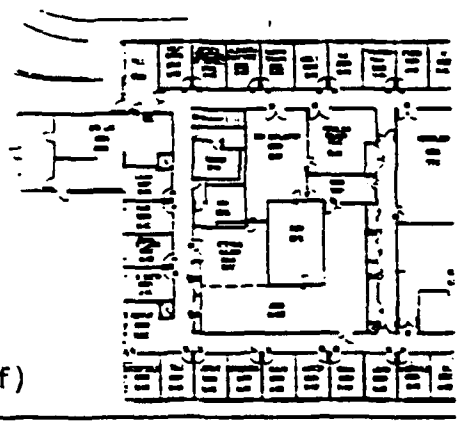
d)

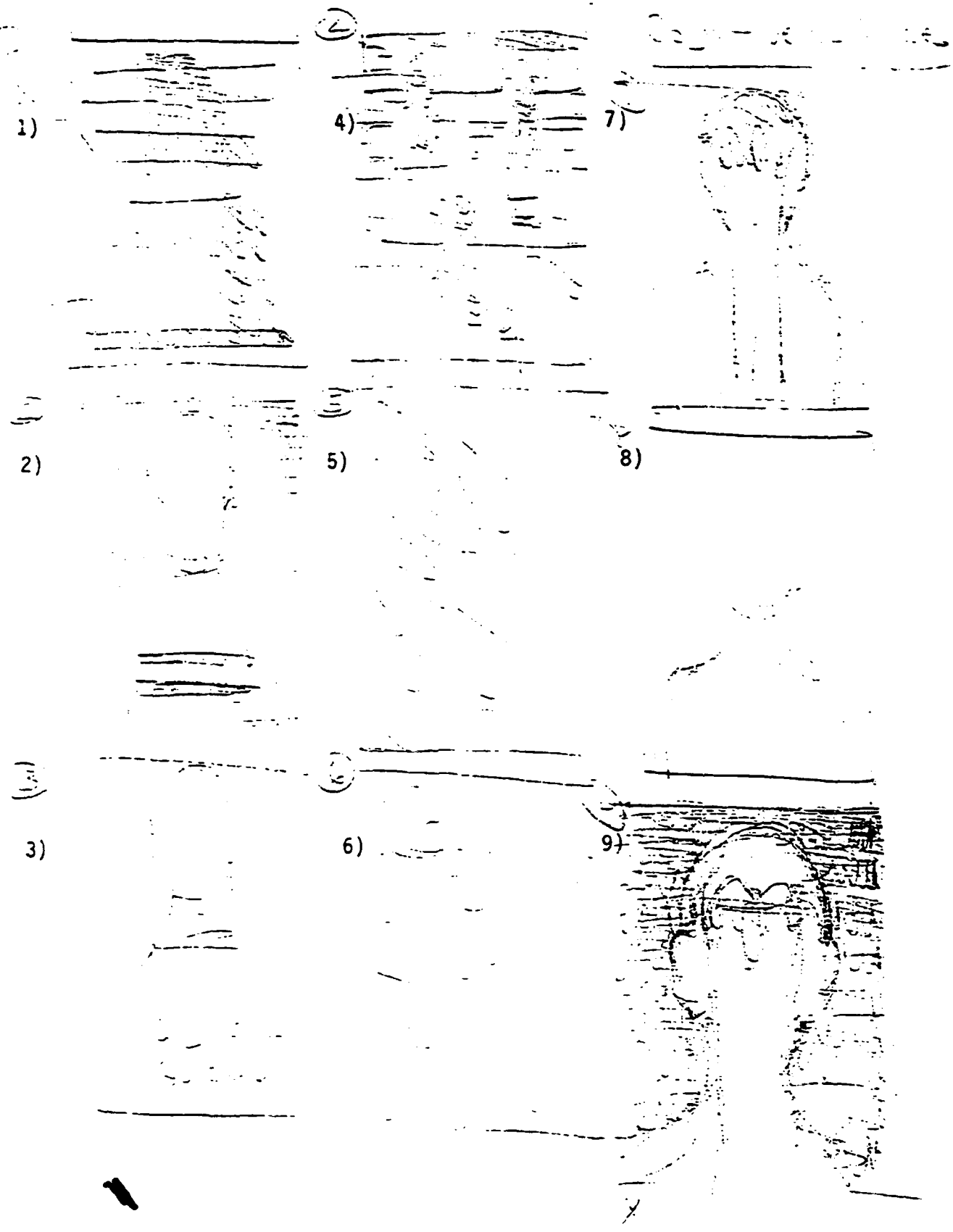


e)

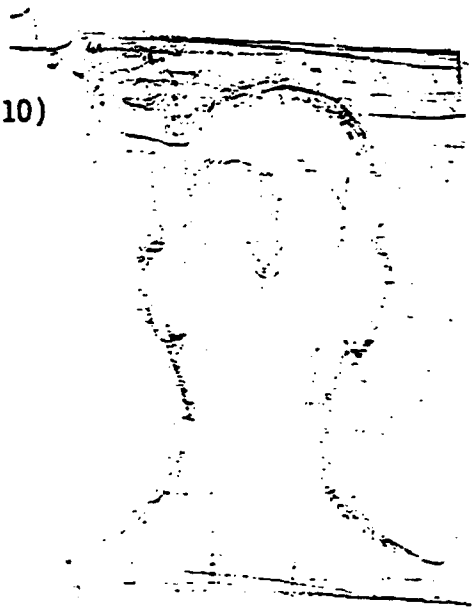


f)




1)
2)
3)
4)
5)
6)
7)
8)
9)

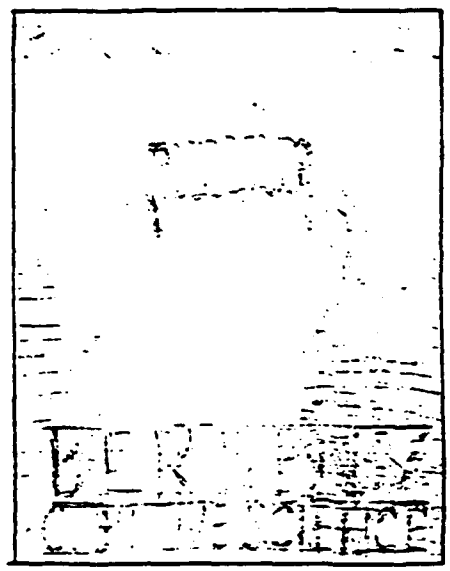
10)



11)

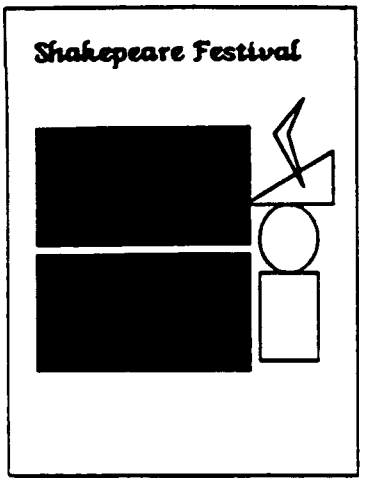
12)

13)

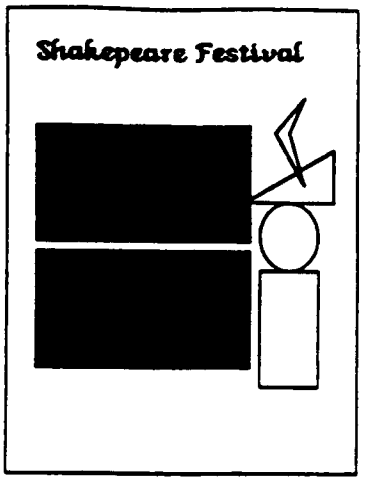


14)

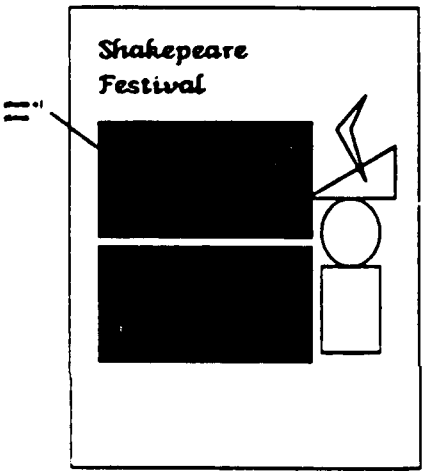
1)


Shakepeare Festival

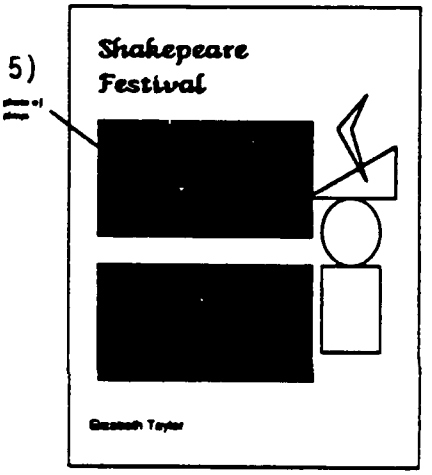
4)


Shakepeare Festival

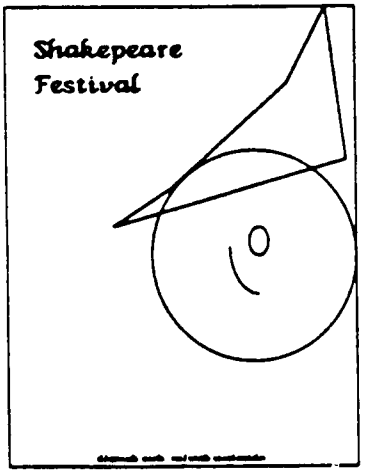
2)


Shakepeare
Festival

5)


Shakepeare
Festival

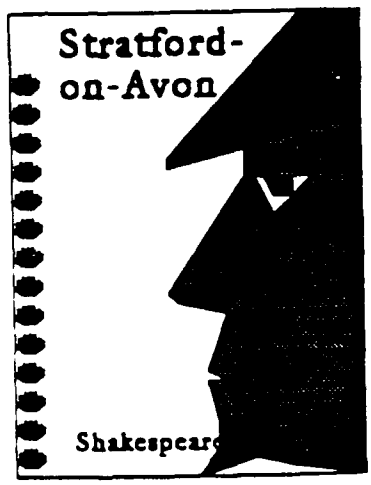
3)


Shakepeare
Festival

6)


Experience
the Experience

7)


Stratford-
on-Avon
Shakespear

8)


Stratford-on-Avon

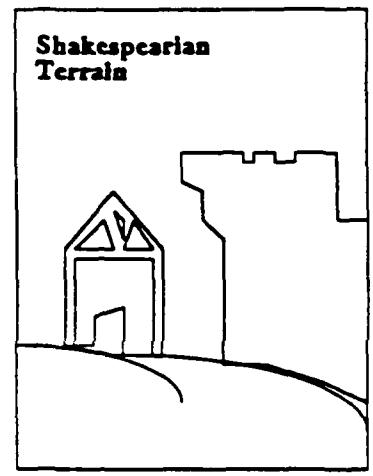
9)


Shakespearian
Terrain

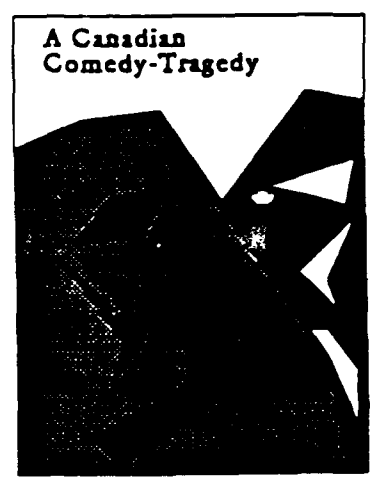
10)

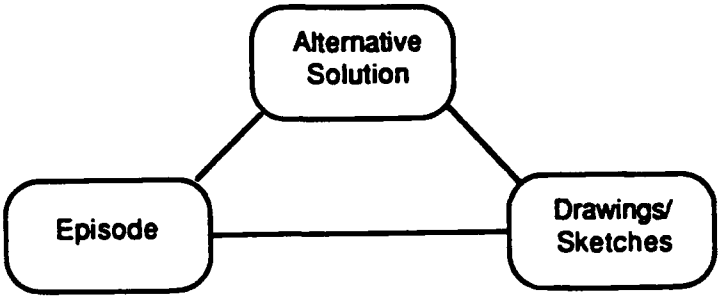

Stratford-
on-Avon

11)


Shakespearian
Terrain

12)


A Canadian
Comedy-Tragedy


Alternative Solution
Episode
Drawings/ Sketches

<code>

where

<code> ::=	<source type> <transformation type> #_of_reinterpretations
<source type> ::=	<syntactic source> <semantic source>
<syntactic source> ::=	LTM <previous_solution>
<semantic source> ::=	LTM <previous_solution>
<previous_solution> ::=	variation identical+/- identical
<transformation type> ::=	<syntactic transformation> <semantic transformation>
<syntactic transformation> ::=	new_generation <transform previous solution>
<semantic transformation> ::=	new_generation <transform previous solution>
<transform previous solution> ::=	lateral_transformation vertical_transformation duplication

Dr. Beth Adelson
Department of Psychology
Rutgers University
Camden, NJ 08102

Dr. David L. Alderton, Code 131
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Nancy S. Anderson
Department of Psychology
University of Maryland
College Park, MD 20742

Dr. Stephen J. Andriole, Chairman
College of Information Studies
Drexel University
Philadelphia, PA 19104

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Michael E. Atwood
NYNEX
AI Laboratory
500 Westchester Avenue
White Plains, NY 10604

Dr. Patricia Baggett
School of Education
610 E. University, Rm 1302D
University of Michigan
Ann Arbor, MI 48109-1259

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Isaac I. Bejar
Law School Admissions
Services
Box 40
Newtown, PA 18940-0040

Dr. William O. Berry
Director of Life and
Environmental Sciences
AFOSR/NL, N1, Bldg. 410
Bolling AFB, DC 20332-6448

Dr. Thomas G. Bever
Department of Psychology
University of Rochester
River Station
Rochester, NY 14627

Dr. Menucha Birenbaum
Educational Testing
Service
Princeton, NJ 08541

Dr. Gautam Biswas
Department of Computer Science
Box 1688, Station B
Vanderbilt University
Nashville, TN 37235

Dr. Kenneth R. Boff
AL/CFH
Wright-Patterson AFB
OH 45433-6573

Dr. Jeff Bonar
Guidance Technology, Inc.
800 Vinial Street
Pittsburgh, PA 15212

Dr. Robert Breaux
Code 252
Naval Training Systems Center
Orlando, FL 32826-3224

Dr. John T. Bruer
James S. McDonnell Foundation
Suite 1610
1034 South Brentwood Blvd.
St. Louis, MO 63117

Dr. Richard Catrambone
School of Psychology
Georgia Institute of Technology
Atlanta, GA 30332-0170

Dr. Ruth W. Chabay
CDBC, Hamburg Hall
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Paul R. Chatelier
Perceptronics
1911 North Ft. Myer Dr.
Suite 1100
Arlington, VA 22209

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Susan Chipman
Cognitive Science Program
Office of Naval Research
800 North Quincy St.
Arlington, VA 22217-5000

Dr. Raymond E. Christal
UES LAMP Science Advisor
AL/HRMIL
Brooks AFB, TX 78235

Dr. David E. Clement
Department of Psychology
University of South Carolina
Columbia, SC 29208

Chief of Naval Education and
Training (N-5)
NAS Pensacola, FL 32508

Dr. Paul Cobb
Purdue University
Education Building
W. Lafayette, IN 47907

Dr. Rodney Cocking
NIMH, Basic Behavior and
Cognitive Science Research
5600 Fishers Lane, Rm 11C-10
Parklawn Building
Rockville, MD 20857

Director, Life Sciences
Office of Naval Research
Code 114
Arlington, VA 22217-5000

Director, Cognitive and
Neural Sciences, Code 1142
Office of Naval Research
Arlington, VA 22217-5000

Library, Code 231
Navy Personnel R&D Center
San Diego, CA 92152-5800

Commanding Officer
Naval Research Laboratory
Code 4827
Washington, DC 20375-5000

Dr. Michael Cowen
Code 142
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. William Crano
Department of Psychology
Texas A&M University
College Station, TX 77843

CTB/McGraw-Hill Library
2500 Garden Road
Monterey, CA 93940-5380

Dr. Denise Cummins
Psychology Department
University of Arizona
Tucson, AZ 85721

Dr. Geory Delacote
Exploratorium
3601 Lyon Street
San Francisco, CA
94123

Dr. Sharon Derry
Florida State University
Department of Psychology
Tallahassee, FL 32306

Dr. Stephanie Doane
University of Illinois
Department of Psychology
603 East Daniel Street
Champaign, IL 61820

Dr. J. Stuart Donn
Faculty of Education
University of British Columbia
2125 Main Mall
Vancouver, BC CANADA V6T 1Z4

Dr. Michael Drillings
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Defense Technical
Information Center
DTIC/DDA-2
Cameron Station, Bldg 5
Alexandria, VA 22314
(4 Copies)

Dr. Pierre Duguet
Organization for Economic
Cooperation and Development
2, rue Andre-Pascal
75016 PARIS
FRANCE

Dr. Richard Duran
Graduate School of Education
University of California
Santa Barbara, CA 93106

Dr. Nancy Eldredge
College of Education
Division of Special Education
The University of Arizona
Tucson, AZ 85721

Dr. John Ellis
Navy Personnel R&D Center
Code 15
San Diego, CA 92152-6800

Dr. Susan Embretson
University of Kansas
Psychology Department
426 Fraser
Lawrence, KS 66045

Dr. Susan Epstein
144 S. Mountain Avenue
Montclair, NJ 07042

ERIC Facility Acquisitions
1301 Piccard Drive, Suite 300
Rockville, MD 20850-4305

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Campus Box 345
Boulder, CO 80309-0345

Dr. Lorraine D. Eyde
US Office of Personnel Management
Office of Personnel Research and Development
1900 E St., NW
Washington, DC 20415

Dr. Franco Faina
Direttore Generale LEVADIFE
Piazzale K. Adenauer, 3
00144 ROMA EUR
ITALY

Dr. Beatrice J. Farr
Army Research Institute
PERI-IC
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Marshall J. Farr
Farr-Sight Co.
2520 North Vernon Street
Arlington, VA 22207

Dr. P-A. Federico
Code 51
NPRDC
San Diego, CA 92152-6800

Dr. Richard L. Ferguson
American College Testing
P.O. Box 168
Iowa City, IA 52243

Mr. Wallace Feurzeig
Educational Technology
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02238

Dr. J. D. Fletcher
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Dr. Kenneth D. Forbus
Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201

Dr. Carl H. Frederiksen
Dept. of Educational Psychology
McGill University
3700 McTavish Street
Montreal, Quebec
CANADA H3A 1Y2

Dr. Alfred R. Fregly
AFOSR/NL, Bldg. 410
Bolling AFB, DC 20332-6448

Dr. Merrill F. Garrett
Director of Cognitive Science
Department of Psychology, Room 312
University of Arizona
Tucson, AZ 85721

Dr. Dedre Gentner
Northwestern University
Department of Psychology
2029 Sheridan Road
Swift Hall, Rm 102
Evanston, IL 60208-2710

Chair, Department of Computer Science
George Mason University
Fairfax, VA 22030

Dr. Helen Gigley
Naval Research Lab., Code 5530
4555 Overlook Avenue, S. W.
Washington, DC 20375-5000

Dr. Philip Gillis
ARI-Fort Gordon
ATTN: PERI-ICD
Fort Gordon, GA 30905

Dr. Robert Glaser
Learning Research & Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Marvin D. Glock
101 Homestead Terrace
Ithaca, NY 14856

Dr. Sam Glucksberg
Department of Psychology
Princeton University
Princeton, NJ 08544-1010

Dr. Susan R. Goldman
Peabody College, Box 45
Vanderbilt University
Nashville, TN 37203

Mr. Harold Goldstein
University of DC
Department Civil Engineering
Bldg. 42, Room 112
4200 Connecticut Avenue, N.W.
Washington, DC 20008

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, GA 30332-0205

Jordan Grafman, Ph.D.
Chief, Cognitive Neuroscience Section, Medical Neurology Branch-NINDS, Bldg 10, Rm. 5C422
Bethesda, MD 20892

Dr. Wayne Gray
Graduate School of Education
Fordham University
113 West 60th Street
New York, NY 10023

Dr. Bert Green
Johns Hopkins University
Department of Psychology
Charles & 34th Street
Baltimore, MD 21218

Dr. James G. Greeno
School of Education
Stanford University
Room 311
Stanford, CA 94305

Dr. Stephen Grossberg
Center for Adaptive Systems
Room 244
111 Cummington Street
Boston University
Boston, MA 02215

Naval Command, Control and Ocean Surveillance Center
RDT&E Division
Attn: Mr. J. Grossman, Code 44
Bldg. 334
San Diego, CA 92152-5000

Dr. Michael Haboe
DORNIER GMBH
P.O. Box 1420
D-7990 Friedrichshafen 1
WEST GERMANY

Marilyn Halpern, Librarian
Brigham Library
Educational Testing Service
Carter and Rosedale Roads
Princeton, NJ 08541

Dr. Kristian J. Hammond
Department of Computer Science
University of Chicago
1100 E. 58th St.
Chicago, IL 60637

Dr. Ivar Hansen
Forsvarets Psykologitjeneste
Akershus/Oslo mil
0015 Oslo 1
NORWAY

Dr. Stephen J. Hanson
Learning & Knowledge Acquisition Research
Siemens Research Center
755 College Road East
Princeton, NJ 08540

Stevan Harnad
Editor, The Behavioral and Brain Sciences
20 Nassau Street, Suite 240
Princeton, NJ 08542

Dr. Delwyn Harnisch
University of Illinois
51 Gerty Drive
Champaign, IL 61820

Janice Hart
Department of the Navy
Joint CALS Management Office
5109 Leesburg Pike
Skyline 6, Room 701
Falls Church, VA 22041

Dr. Reid Hastie
University of Colorado
Department of Psychology
Boulder, CO 80309-0344

Dr. Barbara Hayes-Roth
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C
Palo Alto, CA 94304

Dr. James Hiebert
Department of Educational Development
University of Delaware
Newark, DE 19716

Dr. Keith Holyoak
Department of Psychology
University of California
Los Angeles, CA 90024

Prof. Lutz F. Hornke
Institut fur Psychologie
RWTH Aachen
Jaegerstrasse 17/19
D-5100 Aachen
WEST GERMANY

Ms. Julia S. Hough
Cambridge University Press
40 West 20th Street
New York, NY 10011

Dr. William Howell
Chief Scientist
AFHRL/CA
Brooks AFB, TX 78235-5601

Dr. Earl Hunt
Dept. of Psychology, N1-25
University of Washington
Seattle, WA 98195

Dr. Giorgio Ingargiola
Computer Science Department
Temple University
Philadelphia, PA 19122

Dr. Martin J. Ippel
Center for the Study of
Education and Instruction
Leiden University
P. O. Box 9555
2300 RB Leiden
THE NETHERLANDS

Dr. Robert Jannarone
Elec. and Computer Eng. Dept.
University of South Carolina
Columbia, SC 29208

Dr. Robin Jeffries
Hewlett-Packard
Laboratories, 1U-17
P.O. Box 10490
Palo Alto, CA 94303-0969

Dr. Bonnie E. John
Department of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Dr. Edgar M. Johnson
Technical Director
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Daniel B. Jones
US Nuclear Regulatory
Commission
NRR/12 E4
Washington, DC 20555

Dr. John Jonides
Department of Psychology
University of Michigan
Ann Arbor, MI 48104

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Ruth Kanfer
University of Minnesota
Department of Psychology
Elliott Hall
75 E. River Road
Minneapolis, MN 55455

Dr. Michael Kaplan
Office of Basic Research
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. A. Karmiloff-Smith
MRC-CDU
17 Gordon Street
London
ENGLAND WC1H OAH

Dr. Milton S. Katz
PSC 802 Box 15
FPO AE 09499-1500

Dr. Wendy Kellogg
IBM T. J. Watson
Research Ctr.
P.O. Box 704
Yorktown Heights, NY 10598

Dr. Jeffery L. Kennington
Computing Science and Engineering
School of Engr. & Applied Sciences
Southern Methodist University
Dallas, TX 75275

Dr. Jeremy Kilpatrick
Department of
Mathematics Education
105 Aderhold Hall
University of Georgia
Athens, GA 30602

Dr. Susan S. Kirschenbaum
Code 2212, Building 1171/1
Naval Underwater Systems Center
Newport, RI 02841

Dr. Janet L. Kolodner
Georgia Institute of Technology
College of Computing
Atlanta, GA 30332-0280

Dr. Richard J. Koubek
School of Industrial
Engineering
Grissom Hall
Purdue University
West Lafayette, IN 47907

Dr. Patrick Kyllonen
AFHRL/MOEL
Brooks AFB, TX 78235

Dr. M. Diane Langston
ICL North America
11490 Commerce Park Drive
Reston, VA 22091

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Yuh-Jeng Lee
Department of Computer Science
Code CS/LE
Naval Postgraduate School
Monterey, CA 93943

Dr. Paul E. Lehner
Department of Information
Systems & Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030-4444

Dr. Richard Lesh
Educational Testing Service
Princeton, NJ 08541

Dr. Marcia C. Linn
Graduate School
of Education, EMST
Tolman Hall
University of California
Berkeley, CA 94720

Logicon Inc. (Attn: Library)
Tactical and Training Systems
Division
P.O. Box 85158
San Diego, CA 92138-5158

MinR Dr. Christian Lohwasser
Heerespsychologischer Dienst
Maria Theresien-Kaserne
1130 Wien
AUSTRIA

LT(N) C. D. F. Lyon
Command Personnel Applied
Research Coordinate
Maritime Command Headquarters
FMO Halifax, N.S., B3K 2XO
CANADA

Dr. Jane Malin
Mail Code ER22
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy
Code 04
NETPMSA
Pensacola, FL 32509-5000

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Elizabeth Martin
AL/HRA, Stop 44
Williams AFB
AZ 85240

Dr. John Mayer
Technical Communications
Program
2360 Bonisteel Blvd.
University of Michigan
Ann Arbor, MI 48109

Dr. Richard E. Mayer
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. David J. McGuinness
Gallaudet University
800 Florida Avenue, N.E.
Washington, DC 20002

Dr. Joseph McLachlan
Navy Personnel Research
and Development Center
Code 14
San Diego, CA 92152-6800

Dr. Michael McNeese
DET-1, AL\CFHI
BLDG 248
Wright-Patterson AFB, OH 45432

Dr. Stig Meincke, Ph.D.
Forsvarets Center for Lederskab
Christianshavns Voldgade 8
1424 Kobenhavn K
DENMARK

Dr. Alan Meyrowitz
Naval Research Laboratory
Code 5510
4555 Overlook Ave., SW
Washington, DC 20375-5000

Dr. Joel A. Michael
Department of Physiology
Rush Presbyterian-St Lukes
Medical Center
Rush Medical College
Chicago, IL 60612

Dr. Christine M. Mitchell
School of Indus. and Sys. Eng.
Center for Man-Machine
Systems Research
Georgia Institute of Technology
Atlanta, GA 30332-0205

Dr. Ben B. Morgan, Jr.
Department of Psychology
University of Central Florida
Orlando, FL 32816-0150

Dr. Randy Mumaw
Human Sciences
Westinghouse Science
& Technology Ctr.
1310 Beulah Road
Pittsburgh, PA 15235

Chair, Department of Weapons and
Systems Engineering
U.S. Naval Academy
Annapolis, MD 21402

Distribution Support Division
Bureau of Naval Personnel
PERS-471D
Washington, DC 20370

Academic Progs. & Research Branch
Naval Technical Training Command
Code N-62
NAS Memphis (75)
Millington, TN 38054

Deputy Director Manpower,
Personnel and Training Div.
Naval Sea Systems Command
ATTN: Code 04MP 511
Washington, DC 20362

Navy Supply Systems Command
NAVSUP 5512
ATTN: Sandra Borden
Washington, DC 20376-5000

Mr. J. Nelissen
Tweante University
Fac. Bibl. Toegepaste Onderwyskunde
P. O. Box 217
7500 AE Enschede
The NETHERLANDS

Dr. Raymond S. Nickerson
5 Gleason Road
Bedford, MA 01730

Dr. Nils J. Nilsson
Department of Computer Sciences
Stanford University
Stanford, CA 94305-6060

Director
Training Systems Department
NPRDC (Code 14)
San Diego, CA 92152-6800

Library, NPRDC
Code 041
San Diego, CA 92152-6800

Librarian
Naval Center for Applied Research
in Artificial Intelligence
Naval Research Laboratory
Code 5510
Washington, DC 20375-5000

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Dr. Glenn Osga
NOSC, Code 441
San Diego, CA 92152-6800

Dr. Otchoon Park
Army Research Institute
PERI-2
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Roy Pea
Institute for the
Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201

Dr. Ray S. Perez
ARI (PERI-II)
5001 Eisenhower Avenue
Alexandria, VA 22333

C.V. (MD) Dr. Antonio Perri
Capitano ITNMC
Marispers U.D.G. 3° Sez
MINISTERO DIFESA - MARINA
00100 ROMA - ITALY

Dr. Nancy N. Perry
Naval Education and Training
Program Support Activity
Code-047
Building 2435
Pensacola, FL 32509-5000

CDR Frank C. Petho
Naval Postgraduate
School
Code OR/PE
Monterey, CA 93943

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Peter Pirolli
School of Education
University of California
Berkeley, CA 94720

Dr. Martha Polson
Department of Psychology
University of Colorado
Boulder, CO 80309-0344

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309-0344

Psyc Info - CD and M
American Psychological Assoc.
1200 Uhle Street
Arlington, VA 22201

Mr. Peter Purdue (55Pd)
Operations Research
Naval Postgraduate School
Monterey, CA 93943

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Daniel Reisberg
Reed College
Department of Psychology
Portland, OR 97202

Dr. Brian Reiser
Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201-3142

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Edwina L. Rissland
Dept. of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Mr. W. A. Rizzo
Head, Human Factors Division
Naval Training Systems Center
Code 26
12350 Research Parkway
Orlando, FL 32826-3224

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. William B. Rouse
Search Technology, Inc.
4725 Peachtree Corners Circle
Suite 200
Norcross, GA 30092

Dr. Mark Schlager
SRI International
333 Ravenswood Ave.
Room BS-131
Menlo Park, CA 94025

Dr. Astrid Schmidt-Nielsen
HCI Laboratory
Code 5532
Naval Research Laboratory
Washington, DC 20375-5000

Dr. Alan H. Schoenfeld
University of California
Department of Education
Berkeley, CA 94720

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Colleen M. Seifert
Department of Psychology
University of Michigan
330 Packard Road
Ann Arbor, MI 48104

Mr. Robert Semmes
N218 Elliott Hall
Department of Psychology
University of Minnesota
Minneapolis, MN 55455-0344

Dr. Kishore Sengupta
Department of Administrative
Services, Code 54, SE
Naval Post Graduate School
Monterey, CA 93943-5000

Dr. Valerie L. Shalin
Department of Industrial
Engineering
State University of New York
342 Lawrence D. Bell Hall
Buffalo, NY 14260

Mr. Richard J. Shavelson
Graduate School of Education
University of California
Santa Barbara, CA 93106

Dr. Randall Shumaker
Naval Research Laboratory
Code 5500
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Edward Silver
LRDC
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Mr. Joe Silverman
Code 115
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Jan Sinnott
Department of
Computer Science
Towson State University
Towson, MD 21204

Dr. Robert Smillie
Naval Ocean Systems Center
Code 443
San Diego, CA 92152-5000

Dr. Garold Stasser
Department of Psychology
Miami University
104 Benton Hall
Oxford, OH 45056

Dr. James J. Staszewski
Dept. of Psychology
University of South Carolina
Columbia, SC 29210

Dr. Robert J. Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Kurt Steuck
AL/HRTI
Brooks AFB
San Antonio, TX 78235-5601

Dr. William Stout
University of Illinois
Department of Statistics
101 Illini Hall
725 South Wright St.
Champaign, IL 61820

Dean, College of Behavioral
and Social Sciences
University of Maryland,
Baltimore County
Baltimore, MD 21228

Dr. Jerry Vogt
Department of Psychology
St. Norbert College
De Pere, WI 54115-2099

Dr. Jacques Voneche
University of Geneva
Department of Psychology
Geneva
SWITZERLAND 1204

Dr. Ralph Wachter
Office of Naval Research
Code 1133SS
800 North Quincy Street
Arlington, VA 22217

Dr. Douglas Wetzel
Code 15
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. David Wiley
School of Education
and Social Policy
Northwestern University
Evanston, IL 60208

Dr. Merlin C. Wittrock
Graduate School of Education
Univ. of Calif., Los Angeles
Los Angeles, CA 90024

Dr. Kentaro Yamamoto
03-OT
Educational Testing Service
Rosedale Road
Princeton, NJ 08541

Frank R. Yekovich
Dept. of Education
Catholic University
Washington, DC 20064

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550

Prof. Gerard de Zeeuw
Center for Innovation
and Cooperative Technology
Grote Bickerstraat 72
1013 KS Amsterdam
The NETHERLANDS